**UNIT - 4**

**Structured data:**

Structured data typically categorized as quantitative data is highly organized and easily decipherable by [machine learning algorithms](https://www.ibm.com/cloud/learn/machine-learning). [Developed by IBM in 1974](http://ibm.com/blogs/research/2020/06/sql-relational-model-50-years-later/), structured query language (SQL) is the programming language used to manage structured data. By using a [relational (SQL) database](https://www.ibm.com/cloud/learn/relational-databases), business users can quickly input, search and manipulate structured data.

**Pros and cons of structured data**

Examples of structured data include dates, names, addresses, credit card numbers, etc. Their benefits are tied to ease of use and access, while liabilities revolve around data inflexibility:

**Pros**

* **Easily used by machine learning (ML) algorithms:** The specific and organized architecture of structured data eases manipulation and querying of ML data.
* **Easily used by business users:** Structured data does not require an in-depth understanding of different types of data and how they function. With a basic understanding of the topic relative to the data, users can easily access and interpret the data.
* **Accessible by more tools:** Since structured data predates unstructured data, there are more tools available for using and analyzing structured data.

**Cons**

* **Limited usage:** Data with a predefined structure can only be used for its intended purpose, which limits its flexibility and usability.
* **Limited storage options:** Structured data is generally stored in data storage systems with rigid schemas (e.g., “[data warehouses](https://www.ibm.com/cloud/blog/cloud-data-lake-vs-data-warehouse-vs-data-mart)”). Therefore, changes in data requirements necessitate an update of all structured data, which leads to a massive expenditure of time and resources.

**Structured data tools**

* [**OLAP**](https://www.ibm.com/cloud/learn/olap)**:** Performs high-speed, multidimensional data analysis from unified, centralized data stores.
* [**SQLite**](https://sqlite.org/)**:** Implements a self-contained, [serverless](https://www.ibm.com/cloud/learn/serverless" \t "_blank), zero-configuration, transactional relational database engine.
* [**MySQL**](https://cloud.ibm.com/catalog/content/mysql)**:** Embeds data into mass-deployed software, particularly mission-critical, heavy-load production system.
* [**PostgreSQL**](https://www.ibm.com/cloud/learn/postgresql)**:** Supports SQL and JSON querying as well as high-tier programming languages (C/C+, Java, [Python](https://www.ibm.com/cloud/blog/python-vs-r), etc.).

**Use cases for structured data**

* **Customer relationship management (CRM):** CRM software runs structured data through analytical tools to create datasets that reveal customer behavior patterns and trends.
* **Online booking:** Hotel and ticket reservation data (e.g., dates, prices, destinations, etc.) fits the “rows and columns” format indicative of the pre-defined data model.
* **Accounting:** Accounting firms or departments use structured data to process and record financial transactions.

**Unstructured data**

Unstructured data, typically categorized as qualitative data, cannot be processed and analyzed via conventional data tools and methods. Since unstructured data does not have a predefined data model, it is best managed in [non-relational (NoSQL) databases](https://www.ibm.com/cloud/learn/nosql-databases). Another way to manage unstructured data is to use [data lakes](https://www.ibm.com/cloud/architecture/architectures/cloud-data-lake/) to preserve it in raw form.

The importance of unstructured data is rapidly increasing. [Recent projections](https://www.analyticsinsight.net/the-future-of-data-revolution-will-be-unstructured-data/) indicate that unstructured data is over 80% of all enterprise data, while 95% of businesses prioritize unstructured data management.

**Pros and cons of unstructured data**

Examples of unstructured data include text, mobile activity, social media posts, Internet of Things (IoT) sensor data, etc. Their benefits involve advantages in format, speed and storage, while liabilities revolve around expertise and available resources:

**Pros**

* **Native format:** Unstructured data, stored in its native format, remains undefined until needed. Its adaptability increases file formats in the database, which widens the data pool and enables data scientists to prepare and analyze only the data they need.
* **Fast accumulation rates:** Since there is no need to predefine the data, it can be collected quickly and easily.
* **Data lake storage:**Allows for massive storage and pay-as-you-use pricing, which cuts costs and eases scalability.

**Cons**

* **Requires expertise:** Due to its undefined/non-formatted nature, [data science](https://www.ibm.com/cloud/learn/data-science-introduction) expertise is required to prepare and analyze unstructured data. This is beneficial to data analysts but alienates unspecialized business users who may not fully understand specialized data topics or how to utilize their data.
* **Specialized tools:** Specialized tools are required to manipulate unstructured data, which limits product choices for data managers.

**Unstructured data tools**

* [**MongoDB**](https://www.ibm.com/cloud/learn/mongodb)**:** Uses flexible documents to process data for cross-platform applications and services.
* [**DynamoDB**](https://aws.amazon.com/dynamodb/)**:** Delivers single-digit millisecond performance at any scale via built-in security, in-memory caching and backup and restore.
* [**Hadoop**](https://www.ibm.com/cloud/blog/hadoop-vs-spark)**:** Provides distributed processing of large data sets using simple programming models and no formatting requirements.
* [**Azure**](https://www.ibm.com/cloud/architecture/architectures/ibm-cloud-private-azure/)**:** Enables agile cloud computing for creating and managing apps through Microsoft’s data centers.

**Use cases for unstructured data**

* [**Data mining**](https://www.ibm.com/cloud/learn/data-mining)**:** Enables businesses to use unstructured data to identify consumer behavior, product sentiment, and purchasing patterns to better accommodate their customer base.
* [**Predictive data analytics**](https://www.ibm.com/analytics/predictive-analytics)**:**Alert businesses of important activity ahead of time so they can properly plan and accordingly adjust to significant market shifts.
* [**Chatbots**](https://www.ibm.com/cloud/learn/chatbots-explained)**:** Perform text analysis to route customer questions to the appropriate answer sources.

**ROLE OF MACHINE LEARNING:**

Machine learning for IoT can be used to project future trends, detect anomalies, and augment intelligence by ingesting image, video and audio. Why use machine learning for IoT? Machine learning can help demystify the hidden patterns in IoT data by analyzing massive volumes of data using sophisticated algorithms.

**Benefits of machine learning inference for IoT**

Machine learning is a key component of Software AG’s [Cumulocity IoT](https://www.softwareag.com/en_corporate/platform/iot/iot-analytics-platform.html) low-code, self-service IoT platform. The platform comes ready to go with the tools you need for fast results: device connectivity and management, application enablement and integration, as well as streaming analytics, machine learning, and machine learning model deployment. The platform is available on the cloud, on-premises and/or at the edge. Uniquely with Cumulocity IoT, standalone, edge-only solutions are also supported.

**Simplify machine learning model training**

Cumulocity IoT Machine Learning is designed to help you quickly build new machine learning models in an easy manner. AutoML support allows the right machine learning model to be chosen for you based on your data, whether that be operational device data captured on the Cumulocity IoT platform or historical data stored in big data archives.

**Flexibility to use your data science library of choice**

There are a wide variety of data science libraries available (e.g., Tensorflow®, Keras, Scikit-learn) for developing machine learning models. Cumulocity IoT Machine Learning allows models to be developed in data science frameworks of your choice. These models can be transformed into industry-standard formats using open source tools and made available for scoring within Cumulocity IoT.

**Rapid model deployment to operationalize machine learning quickly**

Whether created within Cumulocity IoT Machine Learning itself or imported from other data science frameworks, model deployment into production environments is possible wherever needed in one click, either in the cloud or at the edge. Operationalized models can be easily monitored and updated if underlying patterns shift. Additionally, pretrained and verified models are available for immediate model deployment to accelerate adoption.

**Prebuilt connectors for operational & historical datastores**

Cumulocity IoT Machine Learning provides easy access to data residing in operational and historical datastores for model training. It can retrieve this data on a periodic basis and route it through an automated pipeline to transform the data and train a machine learning model. Data can be hosted on Amazon® S3 or Microsoft® Azure® Data Lake Storage, as well as local data storage, and retrieved using prebuilt Cumulocity IoT DataHub connectors.

**Integration with Cumulocity IoT Streaming Analytics**

Cumulocity IoT Machine Learning enables high-performance scoring of real-time IoT data within Cumulocity IoT Streaming Analytics. Cumulocity IoT Streaming Analytics provides a “Machine Learning” building block in its visual analytics builder that allows the user to invoke a specified machine learning model to score real-time data. This provides a no-code environment to integrate machine learning models with streaming analytics workflows.

**Notebook integration**

Jupyter Notebook, a de facto standard in data science, provides an interactive environment across programming languages. They can be used to prepare and process data, train, deploy and validate machine learning models. This open-source web application is integrated with Cumulocity IoT Machine Learning.

**NoSQL database**

When people use the term “NoSQL database,” they typically use it to refer to any non-relational database. Some say the term “NoSQL” stands for “non SQL” while others say it stands for “not only SQL.” Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables.

**Brief history of NoSQL databases**

NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model in order to avoid data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity.

As storage costs rapidly decreased, the amount of data that applications needed to store and query increased. This data came in all shapes and sizes — [structured, semi-structured,](https://www.mongodb.com/unstructured-data) and [polymorphic](https://www.mongodb.com/developer/how-to/polymorphic-pattern/) — and defining the schema in advance became nearly impossible. NoSQL databases allow developers to store huge amounts of unstructured data, giving them a lot of flexibility.

Additionally, the [Agile Manifesto](https://agilemanifesto.org/) was rising in popularity, and software engineers were rethinking the way they developed software. They were recognizing the need to rapidly adapt to changing requirements. They needed the ability to iterate quickly and make changes throughout their software stack — all the way down to the database. NoSQL databases gave them this flexibility.

Cloud computing also rose in popularity, and developers began using public clouds to host their applications and data. They wanted the ability to distribute data across multiple servers and regions to make their applications resilient, to scale out instead of scale up, and to intelligently geo-place their data. Some NoSQL databases like MongoDB provide these capabilities.

**NoSQL database features**

Each NoSQL database has its own unique features. At a high level, many NoSQL databases have the following features:

* [Flexible schemas](https://docs.mongodb.com/manual/core/data-modeling-introduction/#flexible-schema)
* [Horizontal scaling](https://www.mongodb.com/basics/scaling)
* [Fast queries due to the data model](https://docs.mongodb.com/manual/core/data-modeling-introduction/#document-structure)
* [Ease of use for developers](https://www.mongodb.com/why-use-mongodb)

Check out [What are the Benefits of NoSQL Databases?](https://www.mongodb.com/nosql-explained/nosql-vs-sql#what-are-the-benefits-of-nosql-databases) to learn more about each of the features listed above.

Types of NoSQL databases

Over time, four major [types of NoSQL databases](https://www.mongodb.com/scale/types-of-nosql-databases) emerged: document databases, [key-value databases](https://www.mongodb.com/databases/key-value-database), wide-column stores, and graph databases.

* Document databases store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects.
* Key-value databases are a simpler type of database where each item contains keys and values.
* Wide-column stores store data in tables, rows, and dynamic columns.
* Graph databases store data in nodes and edges. Nodes typically store information about people, places, and things, while edges store information about the relationships between the nodes.

**Hadoop Ecosystem:**

Hadoop is an open-source software framework used for storing and processing large datasets. The Hadoop ecosystem consists of various tools and technologies that work together to provide a comprehensive big data solution.

**Some of the key components of the Hadoop ecosystem include:**

**1. Hadoop Distributed File System (HDFS):** This is a distributed file system that provides high-throughput access to data across multiple nodes.

**2. MapReduce:** This is a programming model used for processing large datasets in parallel across a cluster of nodes.

**3. YARN:** Yet Another Resource Negotiator (YARN) is a resource management system that manages resources across a Hadoop cluster.

**4. Hive:** This is a data warehouse system that provides SQL-like querying capabilities on top of Hadoop.

**5. Pig:** This is a high-level scripting language used for analyzing large datasets.

**6. HBase:** This is a NoSQL database that provides real-time read/write access to large datasets.

**7. Spark:** This is a fast and general-purpose data processing engine that supports in-memory processing and can run on top of Hadoop.

**8. Mahout:** This is a machine learning library that provides various algorithms for data mining and predictive analytics**.**

Overall, the Hadoop ecosystem provides a powerful and flexible platform for storing, processing, and analyzing large datasets.

**Apache Kafka:**

Apache Kafka is a distributed streaming platform used for building real-time data pipelines and streaming applications. It is often used in conjunction with Hadoop to handle data ingestion and processing.

Apache Spark is a fast and powerful data processing engine that can run on top of Hadoop. It provides in-memory processing capabilities and supports various programming languages such as Java, Scala, and Python. Spark is often used for machine learning, graph processing, and real-time analytics.

Apache Kafka is a distributed data store optimized for ingesting and processing streaming data in real-time. Streaming data is data that is continuously generated by thousands of data sources, which typically send the data records in simultaneously.

Apache Kafka is a distributed data store optimized for ingesting and processing streaming data in real-time. Streaming data is data that is continuously generated by thousands of data sources, which typically send the data records in simultaneously.

Apache Kafka is a distributed streaming system that is emerging as the preferred solution for integrating real-time data from multiple stream-producing sources and making that data available to multiple stream-consuming systems concurrently – including Hadoop targets such as HDFS or HBase.

Kafka was named after the German-language author Franz Kafka, who wrote novels and short stories that often explored themes of alienation, bureaucracy, and the struggle of the individual against a powerful and oppressive system.

In a nutshell, Kafka Streams lets you read data in real time from a topic, process that data (such as by filtering, grouping, or aggregating it) and then write the resulting data into another topic or to other systems of record.

**Edge Streaming Analytics**

[Edge](https://www.hpe.com/in/en/what-is/edge-computing.html) streaming analytics is ingesting a continuous data stream as it’s being created on a device to quickly filter and analyze it in real time. Organizations often use this kind of distributed computation system to get immediate decisions on data that is too substantial to transfer quickly to [the cloud](https://www.hpe.com/in/en/what-is/cloud-computing.html).

**Edge streaming analytics do**

Edge streaming analytics is essentially an automated [compute](https://www.hpe.com/in/en/what-is/compute.html) process done relatively close to the sensor or at a gateway nearby. The stream processing engine takes in the data continuously and runs it through an algorithm to make determinations, monitoring for events that might trigger automatic actions or even just deciding how much of the data is valuable enough to send to a central processor for further action or to the cloud or a storage device. Because it analyses data as it’s generated, it decreases latency in decision-making for connected devices.

**Edge streaming analytics in IoT**

Most organizations run a complex and interconnected system of devices, all creating data that can build into a massive glut of information if it’s not continuously processed. By running data through an analytics algorithm as it’s created at the edge of a corporate network, you can gain faster insights to find new ways to improve efficiency, engage customers, and develop new business.

Related HPE Solutions, Products, or Services

[Intelligent Edge Computing and IoT](https://www.hpe.com/in/en/solutions/edge.html)

[Edge Technology Services](https://www.hpe.com/in/en/services/edge-technology.html)

[HPE Ezmeral Hybrid Data & Analytics Software Platform](https://www.hpe.com/in/en/software.html)

**Why are edge streaming analytics needed**

As the on-demand economy continues to urge companies to deliver more quickly than ever, businesses need to deliver better services at the point of consumption and avoid any lags caused by using remote data centers or clouds. And as the number of connected devices deployed by organizations increases, the volume of data that needs to be processed is growing too, which can quickly overwhelm central data management systems. Edge analytics help enterprises improve real-time business analytics and facilitate faster decisions when time is critical.

Related Topics

[Edge Computing](https://www.hpe.com/in/en/what-is/edge-computing.html)

[Edge to Cloud](https://www.hpe.com/in/en/what-is/edge-to-cloud.html)

[Mobile Edge Computing](https://www.hpe.com/in/en/what-is/mobile-edge-computing.html)

[Edge Datacenter](https://www.hpe.com/in/en/what-is/edge-datacenter.html)

**Benefits of edge streaming analytics**

As organizations create smarter buildings, cities, workspaces, retail experiences, factory floors, and more, there is a huge opportunity to use the information gleaned from all the connected objects—and in real-time. The benefits of edge streaming analytics are many, including:

**Improved uptime**

Because the data is processed on-site rather than being transmitted to a far-off central location, and because enterprise IT is able to look at hardware performance data constantly, it can help organizations develop the foresight to predict and head off failures and avoid unplanned downtime.

**Speed**

Sensors can automatically shut down a machine or take corrective action when a repair is needed. Edge streaming analytics can also speed information to the team to fix it, rather than sending the alert to a central processing location first. And in scientific or engineering enterprises, the rapid-fire generation of real information can accelerate innovation and human progress.

**Scalability**

Because the computational workload is handled at each device, the overall burden is shared across the ecosystem so it can be processed much more efficiently.

**Cost**

By distributing the data processing across edge computing infrastructure, an organization can reduce data transmission and storage costs. In addition, by learning about the health and performance of devices in real-time, repairs and maintenance costs can be tailored to need rather than a broader schedule, which leads to lower operational expenses.

**Security**

Because data is processed at the device, it doesn’t need to be transmitted across the network, which exposes it to risk. Raw data never leaves the device that created it.

**Safety**

When even the tiniest error or delay could spell catastrophe, such as in autonomous driving, local oversight, turn by turn, is critical.

**HPE and edge streaming analytics**

As edge workloads and data volumes continue to increase dramatically, many organizations lack the technical depth or expertise to get their edge programs to the next level. HPE experts have been leading innovation strategies for decades and can help you manage the decentralized world of edge with broad and deep experience in analytics and edge/IoT services. We can help you manage network connectivity for all your edge sites and implement solutions to integrate and handle the continuous streams of value hidden in your data.

[HPE GreenLake edge-to-cloud platform](https://www.hpe.com/in/en/greenlake.html) can help enterprises improve real-time business analytics by providing, managing, and securing edge deployments as a service. With HPE GreenLake, rather than trying to control your continually expanding digital ecosystems yourself, you’ll be able to turn over that work to HPE experts who will set you up correctly, manage it all cost-effectively, and ensure everything is done as securely as possible.

**Network Analytics**

Network analytics is the application of [big data](https://www.techtarget.com/searchdatamanagement/definition/big-data) principles and tools to the data used to manage and secure data networks.

Network analytics provides deeper insight into how the network is performing and how an organization is using the network. IT can use analytics to improve security, fine-tune performance, troubleshoot subtle problems, predict traffic trends, spot potential trouble, and perform deep forensic investigations and audits.

Network analytics is most useful to organizations with complex networks, overtaxed networks or high-level security requirements. As a result, large enterprises are more likely than smaller companies to use network analytics broadly. As the tools become easier to use, more standardized, less expensive or embedded in more managed services, companies with smaller networks will find it more approachable to use analytics. But companies of every size consume network analytics as a built-in feature of cloud-managed network services, such as a [network-as-a-service offering](https://www.techtarget.com/searchnetworking/tip/The-migration-to-enterprise-network-as-a-service), managed software-defined WAN or managed wireless LAN service.

**Xively Cloud for IoT:**

Xively is a system for deploying IoT applications on the cloud. It is offered as PaaS. Xively is basically a data collection, management, and distribution infrastructure. It also provides APIs to connect and develop IoT applications.

Xively comes under the category of a Connected Product Management (CPM) platform. Xively also provides tools to model your connected business. This capability is highly beneficial to build any IoT-based product. Other than that Xively also provides Management as well as operational tools. Xively can be connected to most of the IoT frameworks and microcontrollers in the market to create a ‘smart’ project or product.

Xively [Python](https://www.it4nextgen.com/8-best-full-stack-python-frameworks-to-embed-python-code-to-the-web/) Libraries can also be used to embed python code as per the Xively APIs. A Xively Web interface is provided to be used for easy implementation of the front-end interface. Xively also comes with multiple languages and platform support. We can implement HTTP protocols, APIs, MQTT. This makes device connectivity a lot easier with Xively cloud

All the devices can be connected to [Xively Cloud](https://xively.com/) for real-time processing and archiving to the cloud. IoT application developers can write the front end for IoT applications as per their requirements. This helps in the convenient management of apps with Xively cloud and other APIs. Xively is very popular with companies that deal with IoT-based device manufacturing and development. Companies using Xively can rely on the secure connectivity of devices as well as the seamless data management capability.

**Uses of Xively:**

* Programmers or Developers have to register with Xively to use cloud services.
* After registration and account creation, developers can create different devices for which he has to create an IoT app. It can be easily done using the templates provided in the Web Interface of Xively.
* Each connected device is allocated a unique FEED\_ID. It specifies the data stream and metadata of the connected device.
* Once this is done permissions on the IoT devices are assigned using the available APIs. The available permissions are Create, Update, Delete, and Read.
* One or more bidirectional channels are created after we connect a device with Xively. Each channel is unique to the device connected.
* Xively cloud is connected with the help of these channels.
* Xively APIs are used by IoT devices to create communication-enabled products.

**Python Web Application Framework:**

Python Web framework is a collection of packages or modules that allow developers to write Web applications or services. With it, developers don’t need to handle low-level details like protocols, sockets or process/thread management.

Python web framework will help you with:

* Interpreting requests (getting form parameters, handling cookies and sessions)
* Producing responses (presenting data as HTML or in other formats)
* Storing data persistently (and other things)

Now, let’s look at the most useful and famous Python web framework to help you with Web development.

**PYTHON FULL-STACK FRAMEWORKS:**

A full-stack framework in Python is one which attempts to provide a complete solution for applications. It attempts to supply components for each layer in the stack.

**a. Django**

Django Python is a framework for perfectionists with deadlines. With it, you can build better Web apps in much less time, and in less code. Django is known for how it focusses on automating. It also believes in the DRY (Don’t Repeat Yourself) principle.

Django was originally developed for content-management systems, but is now used for many kinds of web applications. This is because of its templating, automatic database generation, DB access layer, and automatic admin interface generation. It also provides a web server for development use.

Giant companies that use Django Python are- Instagram, Pinterest, Disqus, Mozilla, The Washington Times, and Bitbucket. In fact, when we think of the terms ‘framework’ and ‘Python’, the first thing that comes to our minds is Django.  
We will see more on Django in another lesson.

**b. TurboGears**



With TurboGears, you can create a database-driven, ready-to-extend application in just a few minutes.

It is an MVC web framework with ORM with real multi-database support and support for horizontal data partitioning. It also has a widget system to simplify the development of AJAX apps. You may additionally install its template engine Kajiki.

TurboGears is a microframework and a full-stack solution. It’s PyPI package is called tg.devtools.

**c. web2py**



With web2py, you can develop, deploy, debug, test, administer the database, and maintain applications via the provided web interface. It has no configuration files, and you can even run it off a USB drive.

web2py uses the MVC built-in ticketing system to manage errors.

**d. CubicWeb**

CubicWeb is a semantic web application framework that features a query language and a selection+view mechanism. It also features multiple databases, security, workflows, and reusable components.

**e. Django-hotsauce**

Django-hotsauce is a general-purpose web toolkit that sits on top of Django and other frameworks. It is an interactive Pythonic API that will let you create scalable web applications using the WSGI 1.0 spec. It also provides native bindings for the Schevo DBMS, Durus, ZODB, and Authkit projects

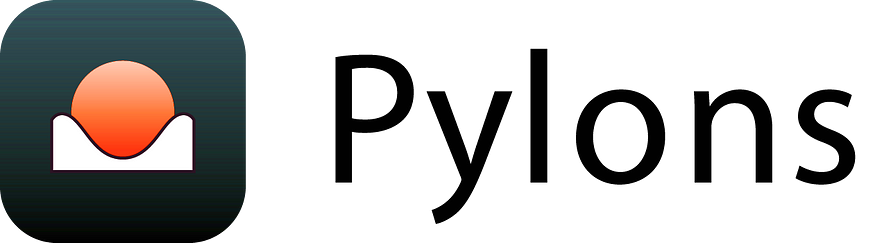
**f. Giotto**

A strict MVC framework that strictly separates Model, View and Controller elements, Giotto makes sure that designers, Web developers, and sysadmins can work independently. It also includes controller modules that allow you to build applications on top of the web, irc or the command line.These are all the most popular Python web framework.

**g. Grok**

Grok was built on the existing Zope 3 libraries. It aims to provide an easier learning curve, and a more agile development experience by emphasizing on convention over configuration and DRY (Don’t Repeat Yourself).

**h. Pylons**



Pylons is a lightweight Web framework aiming at flexibility and rapid development. With the best ideas from Ruby, Python, and Perl, it makes for a structured, but extremely flexible Python Web framework. With Pylons, Web development is fast, flexible, and easy. Pylons is built on top of Paste. But after being merged with Pyramid to form the Pylons project, it is in maintenance-only status.

**i. Reahl**

You can use Reahl to develop web applications in pure Python. However, you may use, customize, or compose widgets in usual Python code. These widgets portray certain server-side and client-side behaviors.

**j. wheezy.web**

Wheezy is a lightweight, high performance, and high concurrency WSGI web framework. Its key features include routing, model update/validation, authentication/authorization, content caching with dependency, middleware, and more. With these, we can build modern, efficient web.

**k. Zope2**



Zope2 is rightly the granddaddy of Python web frameworks, it has been a family of networks. It is a web framework and a general-purpose application server. Today, it is primarily used for CMS. We also have Zope3, which is a standalone framework and a collection of related libraries.

l. Tornado



While Tornado isn’t that famous, it is great with non-blocking I/O. You can scale it to handle tens of thousands of open connections. It makes for a perfect framework for long polling, WebSockets, and other usages needing a continuous connection. Officially, Tornado only supports Linux and BSD OS (Windows and Mac OS X- only for development). Tornado finds its origin in the FriendFeed project, which now belongs to Facebook.

**NON-FULL-STACK FRAMEWORKS IN PYTHON**

A Python non full-stack framework will provide the base application server. This either runs as its own independent process, upon Apache, or in other environments. Let’s look at the most popular ones.

**a. Python Bottle**

Bottle is a simple and fast microframework that you can use to create small Web applications. It provides request-dispatching routes with URL-parameter support, templates, key/value databases, and a built-in HTTP server. It also offers adapters for third-party WSGI/HTTP-server and template engines. This is all in a single file; there are no dependencies except the Python Standard Library.

**b. CherryPy**



It is a pythonic, object-oriented HTTP framework. A web application powered by CherryPy is a standalone Python application that embeds its own multi-threaded web server.  
In a way, CherryPy is a way between the programmer and the problem. It also supports various web servers like Apache, IIS, and so. CherryPy will let you launch multiple HTTP servers at once.

**c. Python Flask**



Like we’ve said before, Flask is a micro framework for Python. It includes a built-in development server, and unit-testing support. It is also fully Unicode-enabled with RESTful request-dispatching and WSGI compliance.

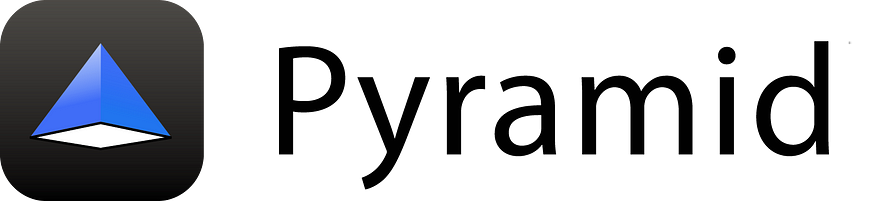
Flask will be useful when you want to develop small, simple applications. With it, you can operate your database however you like- using SQLAlchemy or whatever. A goof Flask example is it is used by LinkedIn and Pinterest.

**d. Hug**



Hug is among the fastest web frameworks for Python. With it, you can build APIs. It supports several API versions, automatic API documentation, and annotation-powered validation. It is built on top of another JSON framework, Falcon.

**e. Pyramid**



Unlike a few that we discussed so far, Pyramid is a framework for large applications. It is flexible; a Pyramid web application starts from a single-file module, and evolves into an ambitious project. You can say that it makes real-world Web application development and deployment more fun, predictable, and productive. Actually, Pyramid is a Pylons project.

**f. Albatross**

It is a small, flexible Python toolkit that lets you develop highly stateful Web applications. Albatross deploys to CGI, FastCGI, and ModPython servers.

**g. Circuits**

Circuits are much like CherryPy, but is a highly efficient web framework to develop standalone multiprocess applications. It supports concurrency, asynchronous I/O components, and is event-driven.

**h. Falcon**



A microframework for small applications, app backends, and higher-level frameworks, Falcon encourages to follow the concept of REST. It is among the fastest web frameworks for Python and is used by EMC, Hurricane Electric, OpenStack, Opera Software, Wargaming, and others.

**i. Growler**



Growler is built on top of asyncio, and is inspired by Connect and Express frameworks for Node.js. If you want ORM or templating, you must install it manually. It handles requests by passing through a middleware chain.

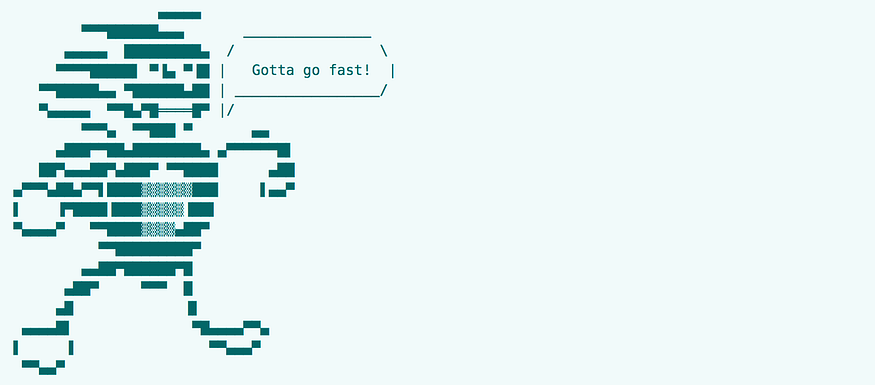
**j. MorePath**

MorePath is a flexible, model-driven web framework. It supports REST and focusses on reusability and extensibility.

**k. Pycnic**

Pycnic is among the fastest web frameworks for Python for developing JSON APIs. The framework is object-oriented and optimized for JSON APIs. It only includes tools for creating Web APIs that leave a lighter footprint.

**l. Sanic**



Sanic is a flask-like framework, but it is fast. It supports asynchronous request handlers, and makes code non-blocking and speedy.

**Django:**

Django provides a unified API to a database backend. Web applications built with Django can work with different databases without requiring any code changes. Django consists of an object-relational mapper, a web templating system and a regular expression based URL dispatcher.

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

**Complete**

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and [up-to-date documentation](https://docs.djangoproject.com/en/stable/).

**Versatile**

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, and XML).

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a [cryptographic hash function](https://en.wikipedia.org/wiki/Cryptographic_hash_function). Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and [clickjacking](https://developer.mozilla.org/en-US/docs/Glossary/Clickjacking) (see [Website security](https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Website_security) for more details of such attacks).

**Scalable**

Django uses a component-based "[shared-nothing](https://en.wikipedia.org/wiki/Shared_nothing_architecture)" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

**Maintainable**

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the [Model View Controller (MVC)](https://developer.mozilla.org/en-US/docs/Glossary/MVC) pattern).

**Portable**

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavors of Linux, Windows, and macOS. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

**AWS FOR IOT**

AWS IoT provides the cloud services that connect your IoT devices to other devices and AWS cloud services. AWS IoT provides device software that can help you integrate your IoT devices into AWS IoT-based solutions. If your devices can connect to AWS IoT, AWS IoT can connect them to the cloud services that AWS provides.

AWS IoT lets you select the most appropriate and up-to-date technologies for your solution. To help you manage and support your IoT devices in the field, AWS IoT Core supports these protocols:

* [MQTT (Message Queuing and Telemetry Transport)](https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html)
* [MQTT over WSS (Websockets Secure)](https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html)
* [HTTPS (Hypertext Transfer Protocol - Secure)](https://docs.aws.amazon.com/iot/latest/developerguide/http.html)
* [LoRaWAN (Long Range Wide Area Network)](https://docs.aws.amazon.com/iot/latest/developerguide/connect-iot-lorawan.html)

The AWS IoT Core message broker supports devices and clients that use MQTT and MQTT over WSS protocols to publish and subscribe to messages. It also supports devices and clients that use the HTTPS protocol to publish messages.

AWS IoT Core for LoRaWAN helps you connect and manage wireless LoRaWAN (low-power long-range Wide Area Network) devices. AWS IoT Core for LoRaWAN replaces the need for you to develop and operate a LoRaWAN Network Server (LNS).

* **AWS IoT Device SDKs**—Build applications on your devices that send messages to and receive messages from AWS IoT. For more information, see [AWS IoT Device SDKs, Mobile SDKs, and AWS IoT Device Client](https://docs.aws.amazon.com/iot/latest/developerguide/iot-sdks.html).
* **AWS IoT Core for LoRaWAN**—Connect and manage your long range WAN (LoRaWAN) devices and gateways by using [AWS IoT Core for LoRaWAN](https://docs.aws.amazon.com/iot/latest/developerguide/connect-iot-lorawan.html).
* **AWS Command Line Interface (AWS CLI)**—Run commands for AWS IoT on Windows, macOS, and Linux. These commands allow you to create and manage thing objects, certificates, rules, jobs, and policies. To get started, see the [AWS Command Line Interface User Guide](https://docs.aws.amazon.com/cli/latest/userguide/). For more information about the commands for AWS IoT, see [iot](https://docs.aws.amazon.com/cli/latest/reference/iot/index.html) in the AWS CLI Command Reference.
* **AWS IoT API**—Build your IoT applications using HTTP or HTTPS requests. These API actions allow you to programmatically create and manage thing objects, certificates, rules, and policies. For more information about the API actions for AWS IoT, see [Actions](https://docs.aws.amazon.com/iot/latest/apireference/API_Operations.html) in the AWS IoT API Reference.
* **AWS SDKs**—Build your IoT applications using language-specific APIs. These SDKs wrap the HTTP/HTTPS API and allow you to program in any of the supported languages.

**NETCONF**

NETCONF works on SSH transport protocol.

• Transport layer provides end-to-end connectivity and ensure reliable delivery of messages.

• NETCONF uses XML-encoded Remote Procedure Calls (RPCs) for framing request and response messages.

• The RPC layer provides mechanism for encoding of RPC calls and notifications.

• NETCONF provides various operations to retrieve and edit configuration data from network devices.

• The Content Layer consists of configuration and state data which is XML-encoded.

• The schema of the configuration and state data is defined in a data modelling language called YANG.

• NETCONF provides a clear separation of the configuration and state data.

• The configuration data resides within a NETCONF configuration data store on the server.

**YANG**

• YANG is a data modelling language used to model configuration and state data manipulated by the NETCONF protocol

• YANG modules contain the definitions of the configuration data, state data, RPC calls that can be issued and the format of the notifications.

• YANG modules defines the data exchanged between the NETCONF client and server.

• A module comprises of a number of 'leaf' nodes which are organized into a hierarchical tree structure.

• The 'leaf' nodes are specified using the 'leaf' or 'leaf-list' constructs.

• Leaf nodes are organized using 'container' or 'list' constructs.

• A YANG module can import definitions from other modules.

• Constraints can be defined on the data nodes, e.g. allowed values.

• YANG can model both configuration data and state data using the 'config' statement.