

What Is Data Engineering and Is It Right for You?

by [Kyle Stratis](#) ⌚ Dec 14, 2020 💬 [8 Comments](#) 🏷️ [basics](#)

Mark as Completed



[Tweet](#)

[Share](#)

[Email](#)

Table of Contents

- [What Do Data Engineers Do?](#)
- [What Are the Responsibilities of Data Engineers?](#)
 - [Data Flow](#)
 - [Data Normalization and Modeling](#)
 - [Data Cleaning](#)
 - [Data Accessibility](#)
- [What Are Common Data Engineering Skills?](#)
 - [General Programming Skills](#)
 - [Database Technologies](#)
 - [Distributed Systems and Cloud Engineering](#)
- [What Isn't Data Engineering?](#)
 - [Data Science](#)
 - [Business Intelligence](#)
 - [Machine Learning Engineering](#)
- [Conclusion](#)

[Remove ads](#)

Big data. Cloud data. AI training data and personally identifying data. Data is all around you and is growing every day. It only makes sense that software engineering has evolved to include **data engineering**, a subdiscipline that focuses directly on the transportation, transformation, and storage of data.

Perhaps you've seen big data job postings and are intrigued by the prospect of handling petabyte-scale data. Maybe you're curious about how [generative adversarial networks](#) create realistic images from underlying data. Maybe you've never even heard of data engineering but are interested in how developers handle the vast amounts of data necessary for most applications today.

No matter which category you fall into, this introductory article is for you. You'll get a broad overview of the field, including what data engineering is and what kind of work it entails.

[Help](#)

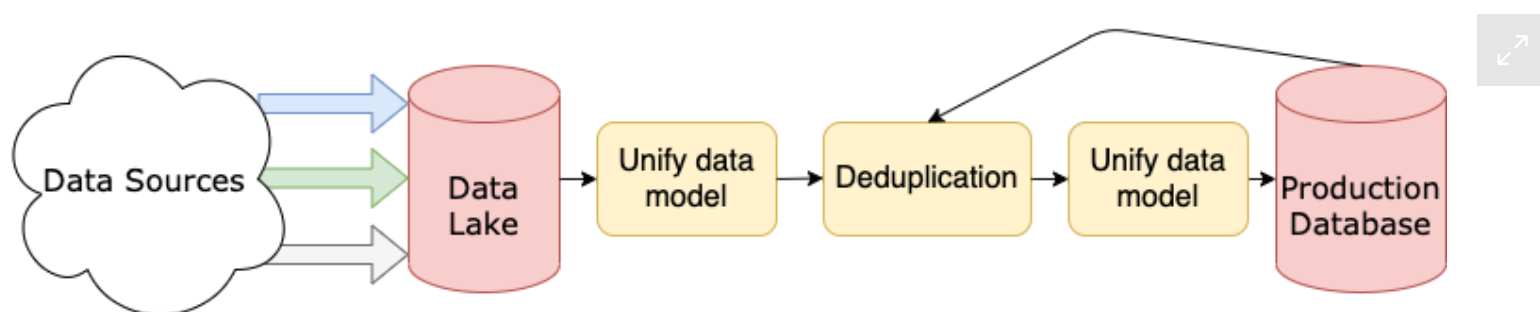
- What the **current state** of the data engineering field is
- How data engineering is **used in the industry**
- Who the various **customers** of data engineers are
- What is and what isn't part of the **data engineering field**
- How to decide if you want to **pursue data engineering** as a discipline

Free Bonus: [Click here to get a Python Cheat Sheet](#) and learn the basics of Python 3, like working with data types, dictionaries, lists, and Python functions.

Data engineering is a very broad discipline that comes with multiple titles. In many organizations, it may not even have a specific title. Because of this, it's probably best to first identify the goals of data engineering and then discuss what kind of work brings about the desired outcomes.

- Training machine learning models
- Doing exploratory data analysis
- Populating fields in an application with outside data

Data pipelines are often distributed across multiple servers:



The data can come from any source:

- [Internet of Things](#) devices
- Vehicle telemetry
- Real estate data feeds
- Normal user activity on a web application
- Any other collection or measurement tools you can think of

The pipeline that the data runs through is the responsibility of the data engineer. Data engineering teams are responsible for the design, construction, maintenance, extension, and often, the infrastructure that supports data pipelines. They may also be responsible for the incoming data or, more often, the [data model](#) and how that data is finally stored.

 Help

Many teams are also moving toward building **data platforms**. In many organizations, it's not enough to have just a single pipeline saving incoming data to an SQL database somewhere. Large organizations have multiple teams that need different levels of access to different kinds of data.

For example, **artificial intelligence (AI)** teams may need ways to label and split cleaned data. **Business intelligence (BI)** teams may need easy access to aggregate data and build data visualizations. **Data science** teams may need database-level access to properly explore the data.

If you're familiar with web development, then you might find this structure similar to the [Model-View-Controller \(MVC\) design pattern](#). With MVC, data engineers are responsible for the model, AI or BI teams work on the views, and all groups collaborate on the controller. Building data platforms that serve all these needs is becoming a major priority in organizations with diverse teams that rely on data access.

Now that you've seen some of what data engineers do and how intertwined they are with the customers they serve, it'll be helpful to learn a bit more about those customers and what responsibilities data engineers have to them.

 [Remove ads](#)

What Are the Responsibilities of Data Engineers?

The customers that rely on data engineers are as diverse as the skills and outputs of the data engineering teams themselves. No matter what field you pursue, your customers will always determine what problems you solve and how you solve them.

In this section, you'll learn about a few common customers of data engineering teams through the lens of their data needs:

- Data science and AI teams
- Business intelligence or analytics teams
- Product teams

Before any of these teams can work effectively, certain needs have to be met. In particular, the data must be:

- Reliably routed into the wider system
- Normalized to a sensible data model
- Cleaned to fill in important gaps
- Made accessible to all relevant to members

These requirements are more fully detailed in the excellent article [The AI Hierarchy of Needs](#) by Monica Rogarty. As a data engineer, you're responsible for addressing your customers' data needs. However, you'll use a variety of approaches to accommodate their individual workflows.

Data Flow

To do anything with data in a system, you must first ensure that it can flow into and through the system reliably. Inputs can be almost any type of data you can imagine, including:

- Live streams of [JSON](#) or XML data
- Batches of videos updated every hour
- Monthly blood-draw data
- Weekly batches of labeled images
- Telemetry from deployed sensors

Data engineers are often responsible for consuming this data, designing a system that can take this data as input from one or many sources, transform it, and then store it for their customers. These systems are often called **ETL** pipelines, which stands for **extract**, **transform**, and **load**.

The data flow responsibility mostly falls under the **extract** step. But the data engineer's responsibility doesn't stop at pulling data into the pipeline. They have to ensure that the pipeline is robust enough to stay up in the face of unexpected or malformed data, sources going offline, and fatal bugs. Uptime is very important, especially when you're consuming live or time-sensitive data.

Your responsibility to maintain data flow will be pretty consistent no matter who your customer is. However, some customers can be more demanding than others, especially when the customer is an application that relies on data being updated in real time.

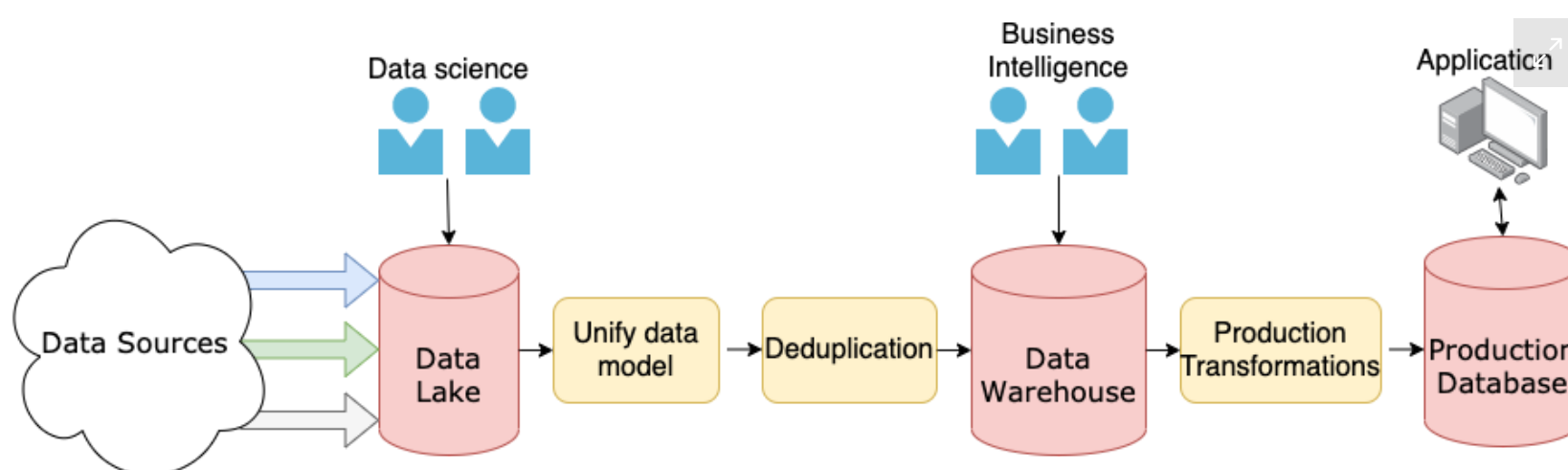
Data Normalization and Modeling

Data flowing into a system is great. However, at some point, the data need to conform to some kind of architectural standard. **Normalizing** data involves tasks that make the data more accessible to users. This includes but is not limited to the following steps:

- Removing duplicates (**deduplication**)
- Fixing conflicting data
- Conforming data to a specified data model

These processes may happen at different stages. For example, imagine you work in a large organization with data scientists and a BI team, both of whom rely on your data. You may store unstructured data in a **data lake** to be used by your data science customers for exploratory data analysis. You may also store the normalized data in a [relational database](#) or a more purpose-built **data warehouse** to be used by the BI team in its reports.

You may have more or fewer customer teams or perhaps an application that consumes your data. The image below shows a modified version of the previous pipeline example, highlighting the different stages at which certain teams may access the data:



In this image, you see a hypothetical data pipeline and the stages at which you'll often find different customer teams working.

If your customer is a product team, then a well-architected data model is crucial. A thoughtful data model can be the difference between a slow, barely responsive application and one that runs as if it already knows what data the user wants to access. These sorts of decisions are often the result of a collaboration between product and data engineering teams.

Data normalization and modeling are usually part of the **transform** step of ETL, but they're not the only ones in this category. Another common transformative step is data cleaning.

[Remove ads](#)

Data Cleaning

Data cleaning goes hand-in-hand with data normalization. Some even consider data normalization to be a subset of data cleaning. But while data normalization is mostly focused on making disparate data conform to some data model, data cleaning includes a number of actions that make the data more uniform and complete, including:

- Casting the same data to a single type (for example, forcing [strings in an integer field to be integers](#))
- Ensuring dates are in the same format
- Filling in missing fields if possible
- Constraining values of a field to a specified range
- Removing corrupt or unusable data

Data cleaning can fit into the deduplication and unifying data model steps in the diagram above. In reality, though each of those steps is very large and can comprise any number of stages and individual processes.

The specific actions you take to clean the data will be highly dependent on the inputs, data model, and desired outcomes. The importance of clean data, though, is constant:

- **Data scientists** need it to perform accurate analyses.
- **Machine learning engineers** need it to build accurate and generalizable models.
- **Business intelligence teams** need it to provide accurate reports and forecasts to the business.
- **Product teams** need it to ensure their product doesn't crash or give faulty information to users.

The data-cleaning responsibility falls on many different shoulders and is dependent on the overall organization and its priorities. As a data engineer, you should strive to automate cleaning as much as possible and do regular spot checks on incoming and stored data. Your customer teams and leadership can provide insight on what constitutes clean data for their purposes.

Data Accessibility

Data accessibility doesn't get as much attention as data normalization and cleaning, but it's arguably one of the more important responsibilities of a customer-centric data engineering team.

Data accessibility refers to how easy the data is for customers to access and understand. This is something that is defined very differently depending on the customer:

- **Data science teams** may simply need data that's accessible with some kind of query language.
- **Analytics teams** may prefer data grouped by some metric, accessible through either basic queries or a reporting interface.
- **Product teams** will often want data that is accessible through fast and straightforward queries that don't change often, with an eye toward product performance and reliability.

Because larger organizations provide these teams and others with the same data, many have moved towards developing their own internal platforms for their disparate teams. A great mature example of this is the ride-hailing service Uber, which has shared many of the details of its [impressive big data platform](#).

In fact, many data engineers are finding themselves becoming platform engineers, making clear the continued importance of data engineering skills to data-driven businesses. Because data accessibility is intimately tied to how data is stored, it's a major component of the **load** step of ETL, which refers to how data is stored for later use.

Now that you've met some common data engineering customers and learned about their needs, it's time to look more closely at what skills you can develop to help address those needs.

What Are Common Data Engineering Skills?

Data engineering skills are largely the same ones you need for software engineering. However, there are a few areas on which data engineers tend to have a greater focus. In this section, you'll learn about several important skill sets:

- General programming concepts
- Databases
- Distributed systems and cloud engineering

Each of these will play a crucial role in making you a well-rounded data engineer.

General Programming Skills

Data engineering is a specialization of software engineering, so it makes sense that the fundamentals of software engineering are at the top of this list. As with other software engineering specializations, data engineers should understand design concepts such as [DRY \(don't repeat yourself\)](#), [object-oriented programming](#), [data structures](#), and algorithms.

As in other specialties, there are also a few favored languages. As of this writing, the ones you see most often in data engineering job descriptions are Python, Scala, and [Java](#). What makes these languages so popular?

Python is popular for several reasons. One of the biggest is its ubiquity. By many measures, Python is among the top three most popular programming languages in the world. For example, it ranked second in the November 2020 [TIOBE Community Index](#) and third in Stack Overflow's [2020 Developer Survey](#).

It's also widely used by machine learning and AI teams. Teams that work closely together often need to be able to communicate in the same language, and Python is still the [lingua franca](#) of the field.

Another, more targeted reason for Python's popularity is its use in orchestration tools like [Apache Airflow](#) and the available libraries for popular tools like [Apache Spark](#). If an organization uses tools like these, then it's essential to know the languages they make use of.

[Scala](#) is also quite popular, and like Python, this is partially due to the popularity of tools that use it, especially Apache Spark. Scala is a functional language that runs on the Java Virtual Machine (JVM), making it able to be used seamlessly with Java.

[Java](#) isn't quite as popular in data engineering, but you'll still see it in quite a few job descriptions. This is partially because of its ubiquity in enterprise software stacks and partially because of its interoperability with Scala. With Scala being used for Apache Spark, it makes sense that some teams make use of Java as well.

In addition to general programming skills, a good familiarity with database technologies is essential.

 [Remove ads](#)

Database Technologies

If you're going to be moving data around, then you're going to be using databases a lot. Very broadly, you can separate database technologies into two categories: SQL and NoSQL.

SQL databases are **relational database management systems** (RDBMS) that model relationships and are interacted with by using Structured Query Language, or SQL. These are commonly used to model data that is defined by relationships, such as customer order data.

Note: If you'd like to learn more about SQL and how to interact with SQL databases in Python, then check out the [Introduction to Python SQL Libraries](#).

NoSQL typically means "everything else." These are databases that usually store nonrelational data, such as the following:

- Key-value stores like [Redis](#) or AWS's [DynamoDB](#)
- Document stores like [MongoDB](#) or [Elasticsearch](#)
- Graph databases like [Neo4j](#)
- Other, less common data stores

While you won't be required to know the ins and outs of all database technologies, you should understand the pros and cons of these different systems and be able to learn one or two of them quickly.

The systems that data engineers work on are increasingly located on the cloud, and data pipelines are usually distributed across multiple servers or clusters, whether on a private cloud or not. Because of this, a prospective data engineer should understand distributed systems and cloud engineering.

Distributed Systems and Cloud Engineering

One of the major advantages of data engineering techniques such as ETL pipelines is that they lend themselves to the implementation of **distributed systems**. A common pattern is to have independent segments of a pipeline running on separate servers orchestrated by a message queue like [RabbitMQ](#) or [Apache Kafka](#).

It's essential to understand how to design these systems, what their benefits and risks are, and when you should use them.

These systems require many servers, and geographically distributed teams often need access to the data they contain. Private cloud providers such as Amazon Web Services, Google Cloud, and Microsoft Azure are extremely popular tools for building and deploying distributed systems.

A basic understanding of the major offerings of cloud providers as well as some of the more popular distributed messaging tools will help you find your first data engineering job. You can expect to learn these tools more in depth on the job.

By now, you’ve learned a lot about what data engineering is. But because there’s no standard definition of the discipline, and because there are a lot of related disciplines, you should also have an idea of what data engineering is *not*.

What Isn’t Data Engineering?

Many fields are closely aligned with data engineering, and your customers will often be members of these fields. It’s important to know your customers, so you should get to know these fields and what separates them from data engineering.

Here are some of the fields that are closely related to data engineering:

- Data science
- Business intelligence
- Machine learning engineering

In this section, you’ll take a closer look at these fields, starting with data science.

Data Science

If data engineering is governed by how you move and organize huge volumes of data, then data science is governed by what you do with that data.

Data scientists commonly query, explore, and try to derive insights from datasets. They may write one-off scripts to use with a specific dataset, while data engineers tend to create reusable programs using software engineering best practices.

Data scientists use statistical tools such as [k-means clustering](#) and [regressions](#) along with machine learning techniques. They often work with R or Python and try to derive insights and predictions from data that will guide decision-making at all levels of a business.

Note: Do you want to explore data science? Take a look at any of the following learning paths:

- [Data Science With Python Core Skills](#)
- [Data Collection & Storage](#)
- [Math for Data Science](#)
- [Pandas for Data Science](#)

Data scientists often come from a scientific or statistical background, and their work style reflects that. They work on a project that answers a specific research question, while a data engineering team focuses on building extensible, reusable, and fast internal products.

A great example of data scientists answering research questions can be found in biotech and health-tech companies, where data scientists explore data on drug interactions, side effects, disease outcomes, and more.

 [Remove ads](#)

Business Intelligence

Business intelligence is similar to data science, with a few important differences. Where data science is focused on forecasting and making future predictions, business intelligence is focused on providing a view of the current state of the business.

Both of these groups are served by data engineering teams and may even work from the same pool of data. Business intelligence, though, is concerned with analyzing business performance and generating reports from the data. These reports then help management make decisions at the business level.

Like data scientists, business intelligence teams rely on data engineers to build the tools that enable them to analyze and report on data relevant to their area of focus.

Machine Learning Engineering

Machine learning engineers are another group you'll come into contact with often. You may do similar work to them, or you might even be embedded in a team of machine learning engineers.

Like data engineers, machine learning engineers are more focused on building reusable software, and many have a computer science background. However, they're less focused on building applications and more focused on building machine learning models or designing new algorithms to be used in models.

Note: If you're interested in the field of machine learning, then check out the [Machine Learning With Python](#) learning path.

The models that machine learning engineers build are often used by product teams in customer-facing products. The data that you provide as a data engineer will be used for training their models, making your work foundational to the capabilities of any machine learning team you work with.

For example, a machine learning engineer may develop a new recommendation algorithm for your company's product, while a data engineer would provide the data used to train and test that algorithm.

One important thing to understand is that the fields you've looked at here often aren't clear-cut. People with a data science, BI, or machine learning background may do data engineering work at an organization, and as a data engineer, you may be called upon to assist these teams in their work.

You could find yourself rearchitected a data model one day, building a data labeling tool another, and optimizing an internal deep learning framework after that. Good data engineers are flexible, curious, and willing to try new things.


Conclusion



That completes your introduction to the field of data engineering, one of the most in-demand disciplines for people with a background or interest in computer science and technology!

In this tutorial, you learned:

- What **data engineers** do
- Who the **customers** of data engineers are
- What **skills** are common to data engineering
- What data engineering **is not**

Now you're at the point where you can decide if you want to go deeper and learn more about this exciting field. Does data engineering sound fascinating to you? Are you interested in exploring it more deeply? Let us know in the comments!


Mark as Completed 

 Python Tricks 

Get a short & sweet **Python Trick** delivered to your inbox every couple of days. No spam ever. Unsubscribe any time. Curated by the Real Python team.

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

[Send Me Python Tricks »](#)

 Help

About **Kyle Stratis**



Kyle is a self-taught developer working as a senior data engineer at Vizia Labs. In the past, he has founded DanqEx (formerly Nasdaq: the original meme stock exchange) and Encryptid Gaming.

[» More about Kyle](#)

Each tutorial at Real Python is created by a team of developers so that it meets our high quality standards. The team members who worked on this tutorial are:



[Aldren](#)



[Geir Arne](#)



[Jon](#)

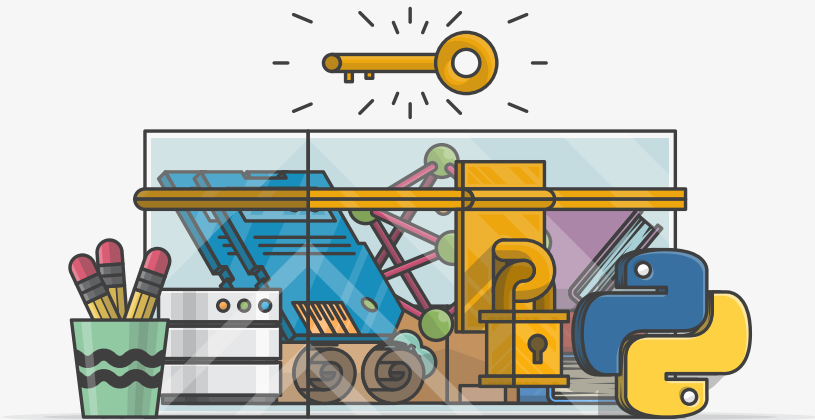


[Joanna](#)



[Jacob](#)

Master Real-World Python Skills
With Unlimited Access to Real Python



Join us and get access to hundreds of tutorials, hands-on video courses, and a community of expert Pythonistas:

[Level Up Your Python Skills »](#)

What Do You Think?

[Tweet](#) [Share](#) [Email](#)

Real Python Comment Policy: The most useful comments are those written with the goal of learning from or helping out other readers—after reading the whole article and all the earlier comments. Complaints and insults generally won’t make the cut here.

[Help](#)

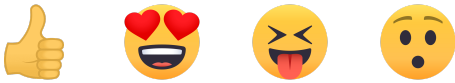
What’s your #1 takeaway or favorite thing you learned? How are you going to put your newfound skills to use? Leave a comment below and let us know.

ALSO ON REAL PYTHON

<div>Python Zip Imports: Distribute Modules ...</div> <div>3 months ago • 4 comm...</div> <div>In this step-by-step tutorial, you'll learn ...</div>	<div>Providing Multiple Constructors in ...</div> <div>a month ago • 3 comme...</div> <div>In this step-by-step tutorial, you'll learn how ...</div>	<div>Python News: What's New From October ...</div> <div>4 months ago • 1 comm...</div> <div>The Python community gave a warm welcome ...</div>	<div>Prettify Your Data Structures With ...</div> <div>4 months ago • 3 comm...</div> <div>The pprint module, Python's data pretty ...</div>
--	--	---	--

What do you think?

16 Responses



8 Comments Real Python Disqus' Privacy Policy Login

Favorite 4 Tweet Share Sort by Best

Join the discussion...

LOG IN WITH OR SIGN UP WITH DISQUS

- Matthias Oberleitner** • a year ago • edited
Overall great and accurate overview. The only thing I would like to add as a Data Engineer is to have a stronger emphasis on SQL Skills.
1 ^ | v • Reply • Share >
- Saul Iversen** • a year ago
This is an excellent overview. I appreciate the effort you put into organizing and communicating your thoughts. It helps me to structure my own as I decide what to pursue as I continue to learn. Thank you.
1 ^ | v • Reply • Share >
- Kyle Stratis** → **Saul Iversen** • a year ago
Thanks for the kind comment :)

Keep Learning

Related Tutorial Categories: basics

serve as a roadmap in this journey, many thanks!!

^ | v • Reply • Share >

Pan Veris • a year ago
Overestimated
I think that D
But this is my

Regards
^ | v • Reply • Share >

AutoSniper • a year ago
Everybody kn
h2oai/datata
them.)
^ | v • Reply • Share >

Andrea Nerli • a year ago
Why people
^ | v • Reply • Share >

Thom • a year ago
I think
^ | v • Reply • Share >

Python Tricks

1 # How to merge two dicts

2 # in Python 3.5+

3

4 >>> x = {'a': 1, 'b': 2}

5 >>> y = {'b': 3, 'c': 4}

6

7 >>> z = {**x, **y}

8

9 >>> z

10 {'c': 4, 'a': 1, 'b': 3}

Get Python Tricks »

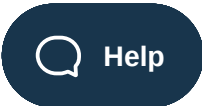
No spam. Unsubscribe any time.

Subscribe Add to Discord

Rapids.AI, not endorsing any of

All Tutorial Topics

- advanced api basics best-practices community databases
- data-science devops django docker flask front-end gamedev
- gui intermediate machine-learning projects python testing



tools

web-dev

web-scraping

Table of Contents

- [What Do Data Engineers Do?](#)
- [What Are the Responsibilities of Data Engineers?](#)
- [What Are Common Data Engineering Skills?](#)
- [What Isn't Data Engineering?](#)
- [Conclusion](#)

Mark as Completed



Tweet



Share



Email

[Remove ads](#)

© 2012–2022 Real Python · [Newsletter](#) · [Podcast](#) · [YouTube](#) · [Twitter](#) · [Facebook](#) · [Instagram](#) · [Python Tutorials](#) · [Search](#) · [Privacy Policy](#) · [Energy Policy](#) · [Advertise](#) · [Contact](#)

Happy Pythoning!



Help