The languages of Data Science

For anyone just getting started on their data science journey, the range of technical options

can be overwhelming. There is a dizzying amount of choice when it comes to programming languages.

Each has it's own strengths and weaknesses and there is no one right answer to the question

of which one you should learn first. The answer to that question depends largely on your needs,

the problems you are trying to solve, and who you are solving them for.

Python, R, and SQL are the languages that we recommend you consider first and foremost.

But there are so many others that have their own strengths and features.

Scala, Java, C++, and Julia are some of the most popular.

Javascript, PHP, Go, Ruby, and Visual Basic all have their own unique use cases as well.

The language you choose to learn will depend on the things you need to accomplish and the

problems you need to solve. It will also depend on what company you work for, what role you

have, and the age of your existing application. We’ll explore the answers to this question

as we dive into the popular languages in the data science industry.

There are many roles available for people who are interested in getting involved in

data science. Business Analyst

Database Engineer Data Analyst

Data Engineer Data Scientist

Research Scientist Software Engineer

Statistician Product Manager

Project Manager and many more.

Let’s dive into what we will learn in Lesson 1. We will put most of our focus on the top

three Data Science languages: Python, R, and SQL, which each have their own lessons. Then

we will go on to highlight other noteworthy languages and what makes them special. Then

we’ll finish with a short quiz to test your knowledge!

In this video, we will review the high-level features of the Python programming language.

Python is a powerhouse language.

It is by far the most popular programming language for data science.

According to the 2019 Kaggle Data Science and Machine Learning Survey, 75% of the over

10,000 respondents from around the world reported that they use Python on a regular basis.

Glassdoor reported that in 2019 more than 75% of data science positions listed included

Python in their job descriptions.

When asked which language an aspiring data scientist should learn first, most data scientists

say Python.

You are probably thinking, why on earth is Python so popular?

Well, let’s start with the people who use Python.

If you already know how to program, then Python is great for you because it uses clear, readable

syntax.

You can do many of the things you are used to doing in other programming languages but

with Python you can do it with less code.

If you want to learn to program, it’s also a great starter language because of the huge

global community and wealth of documentation.

In fact, several different surveys in 2019 found that over 80% of data professionals

worldwide use Python.

Python is useful for many situations, including data science, AI and machine learning, web

development, and IoT devices like the Raspberry Pi.

Large organizations that use Python heavily include IBM, Wikipedia, Google, Yahoo!, CERN,

NASA, Facebook, Amazon, Instagram, Spotify, and Reddit.

Python is a powerful general-purpose programming language that can do a lot of things.

It is widely supported by a global community and shepherded by the Python Software Foundation.

1.

Python is a high-level general-purpose programming language that can be applied to many different

classes of problems.

2.

It has a large, standard library that provides tools suited to many different tasks, including

but not limited to databases, automation, web scraping, text processing, image processing,

machine learning, and data analytics.

3.

For data science, you can use Python's scientific computing libraries such as Pandas, NumPy,

SciPy, and Matplotlib.

4.

For artificial intelligence, it has TensorFlow, PyTorch, Keras, and Scikit-learn.

5.

Python can also be used for Natural Language Processing (NLP) using the Natural Language

Toolkit (NLTK).

Another great selling point is the Python community, which has a well documented history

of paving the way for diversity and inclusion efforts in the tech industry as a whole.

The Python language has a code of conduct executed by the Python Software Foundation

that seeks to ensure safety and inclusion for all, in both online and in person python

communities.

There are also communities like PyLadies that seek to create spaces for people interested

in Python to learn in safe and inclusive environments.

PyLadies is an international mentorship group with a focus on helping more women become

active participants and leaders in the Python open source community.

In this video, we will give a brief overview of the R programming language.

After our last video on Python, where we discussed its wide adoption, you might be wondering

why on earth you should consider learning any other language.

Well, according to the results of the 2019 Kaggle Data Science survey, which had over

10k respondents from around the world, learning up to three languages can increase your salary!

And R has a lot to offer you.

Like Python, R is free to use, but it's a GNU project -- instead of being open source,

it's actually free software.

So if Python is open source and R is free software, what’s the difference?

Well, Both open source and free software commonly refer to the same set of licenses.

Many open source projects use the GNU General Public License, for example.

Both open source and free software support collaboration.

In many cases (but not all), these terms can be used interchangeably.

The Open Source Initiative (OSI) champions open source while the Free Software Foundation

(FSF) defines free software.

Open source is more business focused, while free software is more focused on a set of

values.

Back to why you should learn R. Because this is a free software project, you can use the

language in the same way that you contribute to open source, and it allows for public collaboration

and private and commercial use.

Plus, R is another language supported by a wide global community of people passionate

about making it possible to use the language to solve big problems.

Who is R for?

It's most often used by statisticians, mathematicians, and data miners for developing statistical

software, graphing, and data analysis.

The language’s array-oriented syntax makes it easier to translate from math to code,

especially for someone with no or minimal programming background.

According to Kaggle’s Data Science and Machine Learning Survey, most folks learn R when they're

a few years into their data science career, but it remains a welcoming language to those

who don’t have a software programming background.

R is popular in academia but companies that use R include IBM, Google, Facebook, Microsoft,

Bank of America, Ford, TechCrunch, Uber, and Trulia.

● R has become the world’s largest repository of statistical knowledge.

● As of 2018, R has more than 15,000 publicly released packages, making it possible to conduct

complex exploratory data analysis.

● R integrates well with other computer languages, such as C++, Java, C, .Net, and

Python.

● Common mathematical operations such as matrix multiplication work straight out of

the box.

● R has stronger object-oriented programming facilities than most statistical computing

languages.

There are many ways to connect with other R users around the globe.

Communities such as user!, WhyR?, SatRdays, and R-Ladies are all great to connect with.

And you can also check out the R project website for R conferences and events.

In this video, we'll take a high-level look at SQL.

SQL is a bit different from the other languages we’ve covered so far.

First off, it's formally pronounced “ess cue el,” although some people say “sequel.”

While the acronym stands for “Structured Query Language,” many people do not consider

SQL to be like other software development languages because it's a non-procedural language

and its scope is limited to querying and managing data.

While it is not a “data science” language per se, data scientists regularly use it because

it's simple and powerful!

Another couple of neat facts about SQL: it's much older than Python and R, by about 20

years, having first appeared in 1974.

And, SQL was developed at IBM!

This language is useful in handling structured data; that is, the data incorporating relations

among entities and variables.

SQL was designed for managing data in relational databases.

Here you can see a diagram showing the general structure of a relational database.

A relational database is formed by collections of two-dimensional tables; for example, datasets

and Microsoft Excel spreadsheets.

Each of these tables is then formed by a fixed number of columns and any number of rows.

BUT!

Even though SQL was originally developed for use with relational databases, because it's

so pervasive and easy to use, SQL interfaces for many NoSQL and big data repositories have

also been developed.

The SQL language is subdivided into several language elements, including clauses, expressions,

predicates, queries, and statements.

So what makes SQL great?

Knowing SQL will help you do many different jobs in data science, including business and

data analyst, and it's a must in data engineering.

When performing operations with SQL, you access the data directly.

There's no need to copy it beforehand.

This can speed up workflow executions considerably.

SQL is the interpreter between you and the database.

SQL is an American National Standards Institute, or "ANSI," standard, which means if you learn

SQL and use it with one database, you will be able to easily apply that SQL knowledge

to many other databases.

There are many different SQL databases available, including MySQL, IBM Db2, PostgreSQL, Apache

OpenOffice Base, SQLite, Oracle, MariaDB, Microsoft SQL Server, and more.

The syntax of the SQL you write might change a little bit based on the relational database

management system you’re using.

If you are looking to learn SQL you would be best served to focus on a specific relational

database and then plug into the community for that specific platform.

There are also many great introductory courses on SQL available!

o far, we’ve reviewed Python, R, and SQL.

In this video, we will review some other languages that have compelling use cases for data science.

Ok, so indisputably, Python, R, and SQL are the three most popular languages that data

scientists use.

But, there are many, many other languages that are worth your time when considering

which language to use to solve a particular data science problem.

Scala, Java, C++, and Julia are probably the most traditional data science languages on

this slide.

But JavaScript, PHP, Go, Ruby, Visual Basic, and others have all found their place in the

data science community as well!

I won’t dive as deeply into each of these languages, but I'll mention some notable highlights.

Java is a tried-and-true general-purpose object oriented programming language.

It's been widely adopted in the enterprise space and is designed to be fast and scalable.

Java applications are compiled to bytecode and run on the Java Virtual Machine, or "JVM."

Some notable data science tools built with Java include Weka, for data mining; Java-ML,

which is a machine learning library; Apache MLlib, which makes machine learning scalable;

and Deeplearning4j, for deep learning.

Apache Hadoop is another Java-built application.

It manages data processing and storage for big data applications running in clustered

systems.

Scala is a general-purpose programming language that provides support for functional programming

and a strong static type system.

Many of the design decisions in the construction of the Scala language were made to address

criticisms of Java.

Scala is also interoperable with Java, as it runs on the JVM.

The name "Scala" is a combination of "scalable" and "language."

This language is designed to grow along with the demands of its users.

For data science, the most popular program built using Scala is Apache Spark.

Spark is a fast and general-purpose cluster computing system.

It provides APIs that make parallel jobs easy to write, and an optimized engine that supports

general computation graphs.

Spark includes Shark, which is a query engine; MLlib, for machine learning; GraphX, for graph

processing; and Spark Streaming.

Apache Spark was designed to be faster than Hadoop.

C++ is a general-purpose programming language.

It is an extension of the C programming language, or "C with Classes.”

C++ improves processing speed, enables system programming, and provides broader control

over the software application.

Many organizations that use Python or other high-level languages for data analysis and

exploratory tasks still rely on C++ to develop programs that feed that data to customers

in real-time.

For data science, a popular deep learning library for dataflow called TensorFlow was

built with C++.

But while C++ is the foundation of TensorFlow, it runs on a Python interface, so you don’t

need to know C++ to use it.

MongoDB, a NoSQL database for big data management, was built with C++.

Caffe is a deep learning algorithm repository built with C++, with Python and MATLAB bindings.

A core technology for the World Wide Web, JavaScript is a general-purpose language that

extended beyond the browser with the creation of Node.js and other server-side approaches.

Javascript is NOT related to the Java language.

For data science, the most popular implementation is undoubtedly TensorFlow.js.

TensorFlow.js makes machine learning and deep learning possible in Node.js as well as in

the browser.

TensorFlow.js was also adopted by other open source libraries, including brain.js and machinelearn.js.

The R-js project is another great implementation of JavaScript for data science.

R-js has re-written linear algebra specifications from the R Language into Typescript.

This re-write will provide a foundation for other projects to implement more powerful

math base frameworks like Numpy and SciPy of Python.

Typescript is a superset of JavaScript.

Julia was designed at MIT for high-performance numerical analysis and computational science.

It provides speedy development like Python or R, while producing programs that run as

fast as C or Fortran programs.

Julia is compiled, which means that the code is executed directly on the processor as executable

code; it calls C, Go, Java, MATLAB, R, Fortran, and Python libraries; and has refined parallelism.

The Julia language is relatively new, having been written in 2012, but it has a lot of

promise for future impact on the data science industry.

JuliaDB is a particularly useful application of Julia for data science.

It's a package for working with large persistent data sets.

That's as far as we’ll dig into the many languages that are used to solve data science

problems.

If you have experience with a particular language, I recommend you do a web search to see what

might already be possible in terms of using it for data science.

You might be surprised at the possibilities!

OPEN SOURCE TOOLS

Open source tools are available for various data science tasks.

In this video, we’ll have a look at the different data science tasks.

In subsequent videos we’ll walk through the most commonly used open source tools for

those tasks.

The most important tools are covered throughout this course.

Data Management is the process of persisting and retrieving data.

Data Integration and Transformation, often referred to as Extract, Transform, and Load,

or “ETL,” is the process of retrieving data from remote data management systems.

Transforming data and loading it into a local data management system is also part of Data

Integration and Transformation.

Data Visualization is part of an initial data exploration process, as well as being part

of a final deliverable.

Model Building is the process of creating a machine learning or deep learning model

using an appropriate algorithm with a lot of data.

Model deployment makes such a machine learning or deep learning model available to third-party

applications.

Model monitoring and assessment ensures continuous performance quality checks on the deployed

models.

These checks are for accuracy, fairness, and adversarial robustness.

Code asset management uses versioning and other collaborative features to facilitate

teamwork.

Data asset management brings the same versioning and collaborative components to data.

Data asset management also supports replication, backup, and access right management.

Development environments, commonly known as Integrated Development Environments, or “IDEs”,

are tools that help the data scientist to implement, execute, test, and deploy their

work.

Execution environments are tools where data preprocessing, model training, and deployment

take place.

Finally, there is fully integrated, visual tooling available that covers all the previous

tooling components, either partially or completely.

This concludes this video.

In the next video we’ll start looking at open source tools for data science tasks.

In part one of this two-part series, we’ll cover data management, open source data integration,

transformation, and visualization tools.

The most widely used open source data management tools are relational databases such as

MySQL and PostgreSQL; NoSQL databases such as MongoDB Apache CouchDB, and Apache Cassandra;

and file-based tools such as the Hadoop File System or Cloud File systems like Ceph.

Finally,Elasticsearch is mainly used for storing text data and creating a search index for

fast document retrieval.

The task of data integration and transformation in the classic data warehousing world is called

ETL, which stands for “extract, transform, and load.”

These days, data scientists often propose the term “ELT” – Extract, Load, Transform“ELT”,

stressing the fact that data is dumped somewhere and the data engineer or data scientist themself

is responsible for data.

Another term for this process has now emerged: “data refinery and cleansing.”

Here are the most widely used open source data integration and transformation tools:

Apache AirFlow, originally created by AirBNB; KubeFlow, which enables you to execute data

science pipelines on top of Kubernetes; Apache Kafka, which originated from LinkedIn;

Apache Nifi, which delivers a very nice visual editor;

Apache SparkSQL (which enables you to use ANSI SQL and scales up to compute clusters

of 1000s of nodes), and NodeRED, which also provides a visual editor.

NodeRED consumes so little in resources that it even runs on small devices like a Raspberry

Pi.

We’ll now introduce the most widely used open source data visualization tools.

We have to distinguish between programming libraries where you need to use code and tools

that contain a user interface.

The most popular libraries are covered in the next videos.

A similar approach uses Hue, which can create visualizations from SQL queries.

Kibana, a data exploration and visualization web application, is limited to Elasticsearch

(the data provider).

Finally, Apache Superset is a data exploration and visualization web application.

Model deployment is extremely important.

Once you’ve created a machine learning model capable of predicting some key aspects of

the future, you should make that model consumable by other developers and turn it into an API.

Apache PredictionIO currently only supports Apache Spark ML models for deployment, but

support for all sorts of other libraries is on the roadmap.

Seldon is an interesting product since it supports nearly every framework, including

TensorFlow, Apache SparkML, R, and scikit-learn.

Seldon can run on top of Kubernetes and Redhat OpenShift.

Another way to deploy SparkML models is by using MLeap.

Finally, TensorFlow can serve any of its models using the TensorFlow service.

You can deploy to an embedded device like a Raspberry Pi or a smartphone using TensorFlow

Lite, and even deploy to a web browser using TensorFlow dot JS.

Model monitoring is another crucial step.

Once you’ve deployed a machine learning model, you need to keep track of its prediction

performance as new data arrives in order to maintain outdated models.

Following are some examples of model monitoring tools:

ModelDB is a machine model metadatabase where information about the models are stored and

can be queried.

It natively supports Apache Spark ML Pipelines and scikit-learn.

A generic, multi-purpose tool called Prometheus is also widely used for machine learning model

monitoring, although it’s not specifically made for this purpose.

Model performance is not exclusively measured through accuracy.

Model bias against protected groups like gender or race is also important.

The IBM AI Fairness 360 open source toolkit does exactly this.

It detects and mitigates against bias in machine learning models.

Machine learning models, especially neural-network-based deep learning models, can be subject to adversarial

attacks, where an attacker tries to fool the model with manipulated data or by manipulating

the model itself.

The IBM Adversarial Robustness 360 Toolbox can

be used to detect vulnerability to adversarial attacks and help make the model more robust.

Machine learning modes are often considered to be a black box that applies some mysterious

“magic.”

The IBM AI Explainability 360 Toolkit makes the

machine learning process more understandable by finding similar examples within a dataset

that can be presented to a user for manual comparison.

The IBM AI Explainability 360 Toolkit can also illustrate training for a simpler machine

learning model by explaining how different input variables affect the final decision

of the model.

Options for code asset management tools have been greatly simplified:

For code asset management – also referred to as version management or version control

– Git is now the standard.

Multiple services have emerged to support Git, with the most prominent being GitHub,

which provides hosting for software development version management.

The runner-up is definitely GitLab, which has the advantage of being a fully open source

platform that you can host and manage yourself.

Another choice is Bitbucket.

Data asset management, also known as data governance or data lineage, is another crucial

part of enterprise grade data science.

Data has to be versioned and annotated with metadata.

Apache Atlas is a tool that supports this task.

Another interesting project, ODPi Egeria, is managed through the Linux Foundation and

is an open ecosystem.

It offers a set of open APIs, types, and interchange protocols that metadata repositories use to

share and exchange data.

Finally, Kylo is an open source data lake management software platform that provides

extensive support for a wide range of data asset management tasks.

This concludes part one of this two-part series.

Now let’s move on to part two.

Welcome to part two of this series.

In this section, we’ll cover the development environment, open source data integration,

transformation, and visualization tools.

One of the most popular current development environments that data scientists are using

is “Jupyter.”

Jupyter first emerged as a tool for interactive Python programming; it now supports more than

a hundred different programming languages through “kernels.”

Kernels shouldn’t be confused with operating system kernels.

Jupyter kernels are encapsulating the different interactive interpreters for the different

programming languages.

A key property of Jupyter Notebooks is the ability to unify documentation, code, output

from the code, shell commands, and visualizations into a single document.

JupyterLab is the next generation of Jupyter Notebooks and in the long term, will actually

replace Jupyter Notebooks.

The architectural changes being introduced in JupyterLab makes Jupyter more modern and

modular.

From a user’s perspective, the main difference introduced by JupyterLab is the ability to

open different types of files, including Jupyter Notebooks, data, and terminals.

You can then arrange these files on the canvas.

Although Apache Zeppelin has been fully reimplemented, it’s inspired by Jupyter Notebooks and provides

a similar experience.

One key differentiator is the integrated plotting capability.

In Jupyter Notebooks, you are required to use external libraries in Apache Zeppelin,

and plotting doesn’t require coding.

You can also extend these capabilities by using additional libraries.

RStudio is one of the oldest development environments for statistics and data science, having been

introduced in 2011.

It exclusively runs R and all associated R libraries.

However, Python development is possible and R is therefore tightly integrated into this

tool to provide an optimal user experience.

RStudio unifies programming, execution, debugging, remote data access, data exploration, and

visualization into a single tool.

Spyder tries to mimic the behaviour of RStudio to bring its functionality to the Python world.

Although Spyder does not have the same level of functionality as RStudio, data scientists

do consider it an alternative.

But in the Python world, Jupyter is used more frequently.

This diagram shows how Spyder integrates code, documentation, visualizations, and other components

into a single canvas.

Sometimes your data doesn’t fit into a single computer’s storage or main memory capacity.

That’s where cluster execution environments come in.

The well known cluster-computing framework Apache Spark is among the most active Apache

projects and is used across all industries, including in many Fortune 500 companies.

The key property of Apache Spark is linear scalability.

This means, if you double the number of servers in a cluster, you’ll also roughly double

its performance.

After Apache Spark began to gain market share, Apache Flink was created.

The key difference between Apache Spark and Apache Flink is that Apache Spark is a batch

data processing engine, capable of processing huge amounts of data file by file.

Apache Flink, on the other hand, is a stream processing image, with its main focus on processing

real-time data streams.

Although engine supports both data processing paradigms, Apache Spark is usually the choice

in most use cases.

One of the latest developments in the data science execution environments is called “Ray,”

which has a clear focus on large-scale deep learning model training.

Let’s look at open source tools for data scientists that are fully integrated and visual.

With these tools, no programming knowledge is necessary.

Most important tasks are supported by these tools; these tasks include data integration,

transformation, data visualization, and model building.

KNIME originated at the University of Konstanz in 2004.

As you can see, KNIME has a visual user interface with drag-and-drop capabilities.

It also has built-in visualization capabilities.

Knime can be be extended by programming in R and Python, and has connectors to Apache

Spark.

Another example of this group of tools is Orange.

It’s less flexible than KNIME, but easier to use.

In this video, you’ve learned about the most common data science tasks and which open

source tools are relevant to those tasks.

In the next video, we’ll describe some established commercial tools that you’ll encounter in

your data science experience.

Let’s move on to the next video to get more details.

We previously covered open source tools for data science.

Now, let’s look at the commercial options you’ll find in many enterprise projects.

Let’s revisit our overview of different tool categories.

In data management, most of an enterprise’s relevant data is stored in an

Oracle Database, Microsoft SQL Server, or IBM Db2.

Although open source databases are gaining popularity, those three data management products

are still considered the industry-standard.

They won’t disappear in the near future.

It’s not just about functionality.

Data is at the heart of every organization, and the availability of commercial supports

plays a major role.

Commercial supports are delivered directly from software vendors, influential partners,

and support networks.

When we focus on commercial data integration tools, we’re talking about “extract, transform,

and load,” or “ETL” tools.

According to a Gartner Magic Quadrant, Informatica Powercenter and IBM InfoSphere DataStage are

the leaders, followed by products from SAP, Oracle, SAS, Talend, and Microsoft.

These tools support design and deployment of ETL data-processing pipelines through a

graphical interface.

They also provide connectors to most of the commercial and open source target information

systems.

Finally, Watson Studio Desktop includes a component called Data Refinery, which enables

the defining and execution of data integration processes in a spreadsheet style.

In the commercial environment, data visualizations are utilizing business intelligence, or “BI”,

tools.

Their main focus is to create visually attractive and easy-to-understand reports and live dashboards.

The most prominent commercial examples are: Tableau, Microsoft Power BI, and IBM Cognos

Analytics.

Another type of visualization targets data scientists rather than regular users.

A sample problem might be “How can different columns in a table relate to each other?”

This type of functionality is contained in Watson Studio Desktop.

If you want to build a machine learning model using a commercial tool, you should consider

using a data mining product.

The most prominent of these types of products are: SPSS Modeler and SAS Enterprise Miner.

In addition, A version of SPSS Modeler is also available in Watson Studio Desktop, based

on the cloud version of the tool.

We’ll talk more about cloud-based tools in the next video.

In commercial software, model deployment is tightly integrated in the model building process.

This diagram shows an example of the SPSS Collaboration and Deployment Services which

are used to deploy any type of asset created by the SPSS software tools suite.

Other vendors use the same type of process.

Commercial software can also export models in an open format.

For example, SPSS Modeler supports the exporting of models as Predictive Model Markup Language,

or PMML, which can be read by many other commercial and open software packages.

Model monitoring is a new discipline and there are currently no relevant commercial tools

available.

As a result, open source is the first choice.

The same is true for code asset management.

Open source with Git and GitHub is the effective standard.

Data asset management, often called data governance or data lineage, is a crucial part of enterprise

grade data science.

Data must be versioned and annotated using metadata.

Vendors, including Informatica Enterprise Data Governance and IBM, provide tools for

these specific tasks.

The IBM InfoSphere Information Governance Catalog covers functions like data dictionary,

which facilitates discovery of data assets.

Each data asset is assigned to a data steward -- the data owner.

The data owner is responsible for that data asset and can be contacted.

Data lineage is also covered; this enables a user to track back through the transformation

steps followed in creating the data assets.

The data lineage also includes a reference to the actual source data.

Rules and policies can be added to reflect complex regulatory and business requirements

for data privacy and retention.

Watson Studio is a fully integrated development environment for data scientists.

It’s usually consumed through the cloud, and we’ll cover more about it in a later

lesson.

There is also a desktop version available.

Watson Studio Desktop combines Jupyter Notebooks with graphical tools to maximize data scientists’

performance.

Watson Studio, together with Watson Open Scale, is a fully integrated tool covering the full

data science life cycle and all the tasks we’ve discussed previously.

We’ll talk more about both in the next lesson.

but just keep in mind that they can be deployed in a local data center on top of Kubernetes

or RedHat OpenShift.

Another example of a fully integrated commercial tool is H2O Driverless AI, which covers the

complete data science life cycle.

In this lesson, you’ve learned how most common data science tasks are supported by

commercial tools.

In the next video, we’ll discover data science tools that are available exclusively on the

cloud.