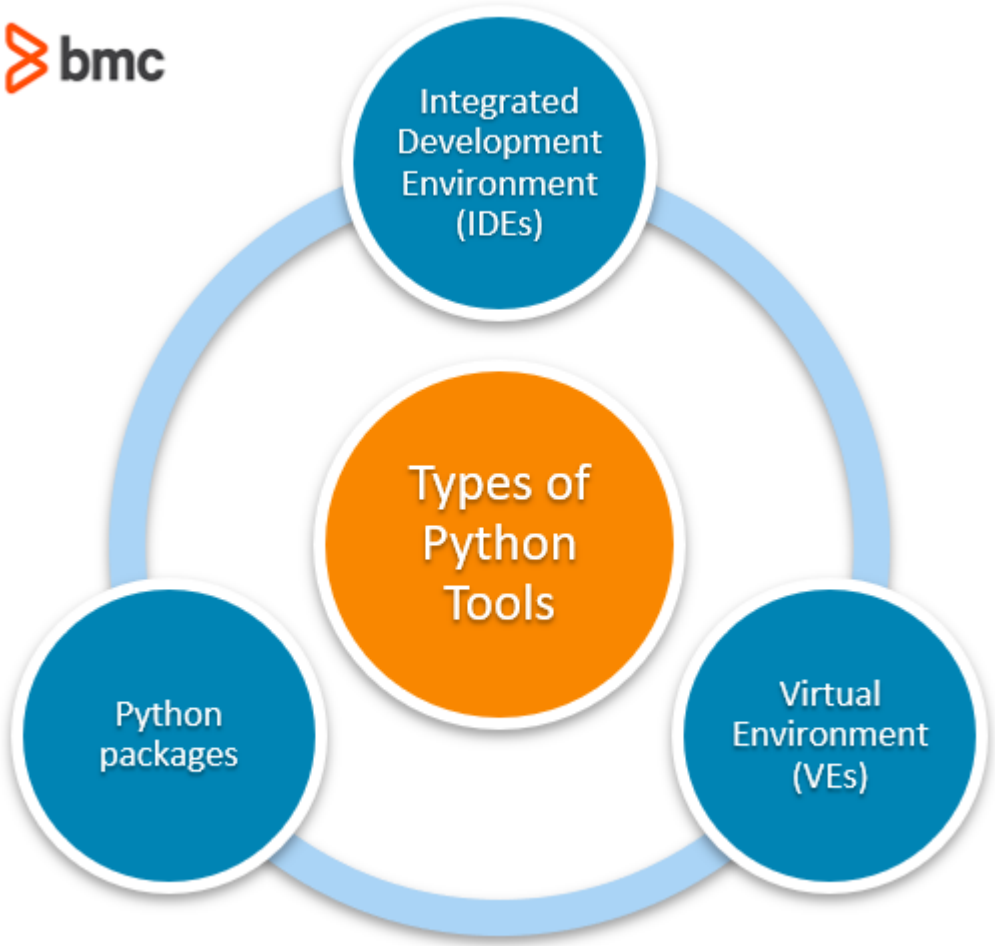
PYTHON DEVELOPMENT TOOLS: YOUR PYTHON STARTER KIT

Every good developer needs a good set of tools. Most of a dev's time will be spent writing inside an IDE. Likely, that dev will undertake multiple projects at the same time, so having a way to organize those projects is very helpful.

Let's take a look at some Python development tools—since Python is one of the most popular languages globally. The tools you choose will depend on the purpose of your project as well as dependencies and system requirements. That's why we'll look at three types of tools you'll use frequently:



IDEs

Right out of the gate, your IDE will be important—that is, your integrated development environment. IDEs tend to be chosen by a developer's taste, but some are better-suited than others for specific purposes.

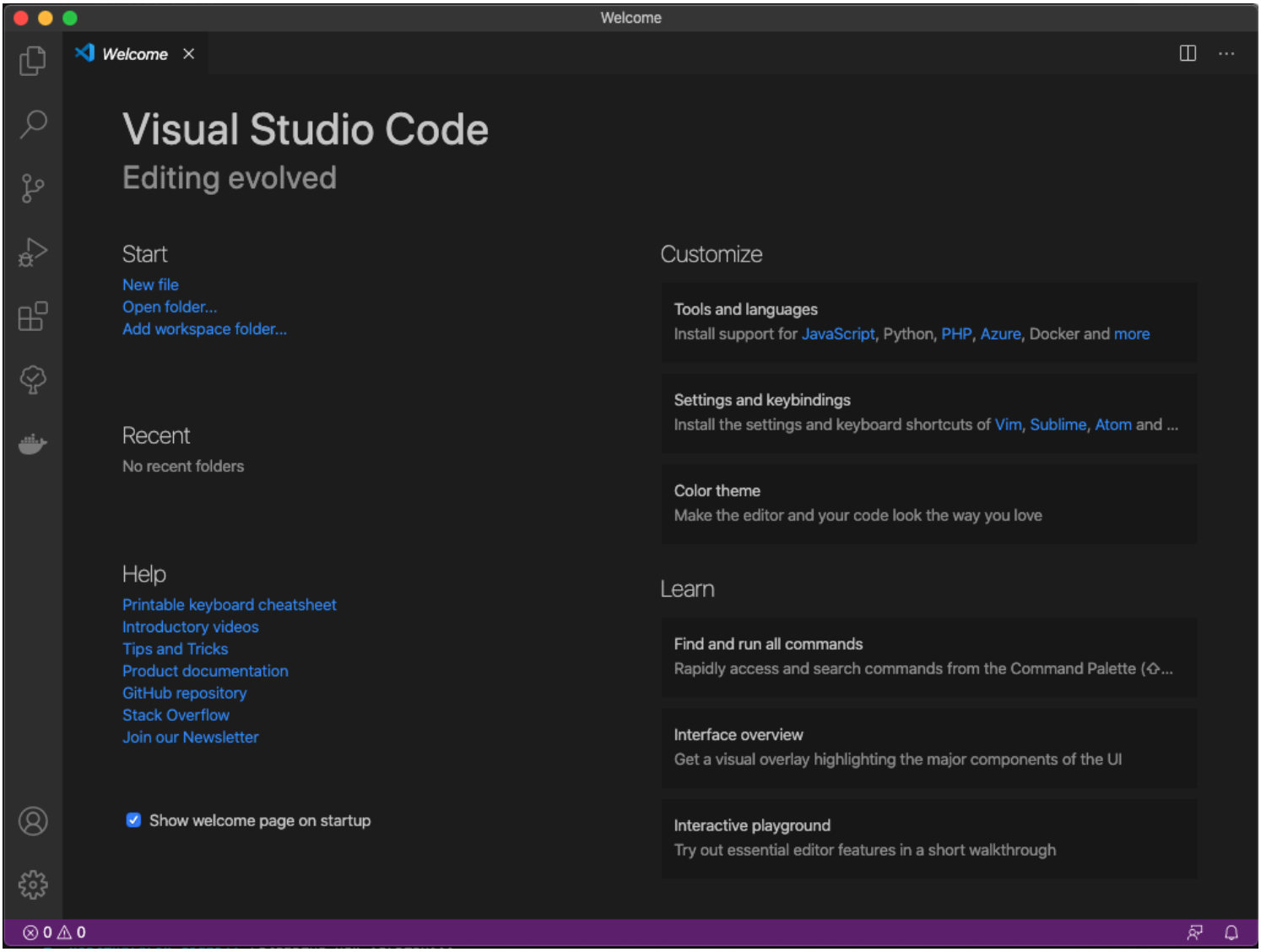
Switching between one or another is very easy. It's as simple as opening a text document in Microsoft Word, or Notepad, so selecting which to use is not a heavy choice. Likely, you will bounce around between a couple until you settle on something that works for you.

Here are some of the most popular Python IDEs, which we'll explore below:

- VS Code
- Atom
- PyCharm
- Spyder

Jupyter notebooks are useful to share and read code. Formerly known as iPython notebooks, Jupyter notebooks are a file type that can be read using Jupyter itself, or by some IDEs. VSCode, Atom, Pycharm, and Google's Colaboratory notebooks all read and run Jupyter notebooks.

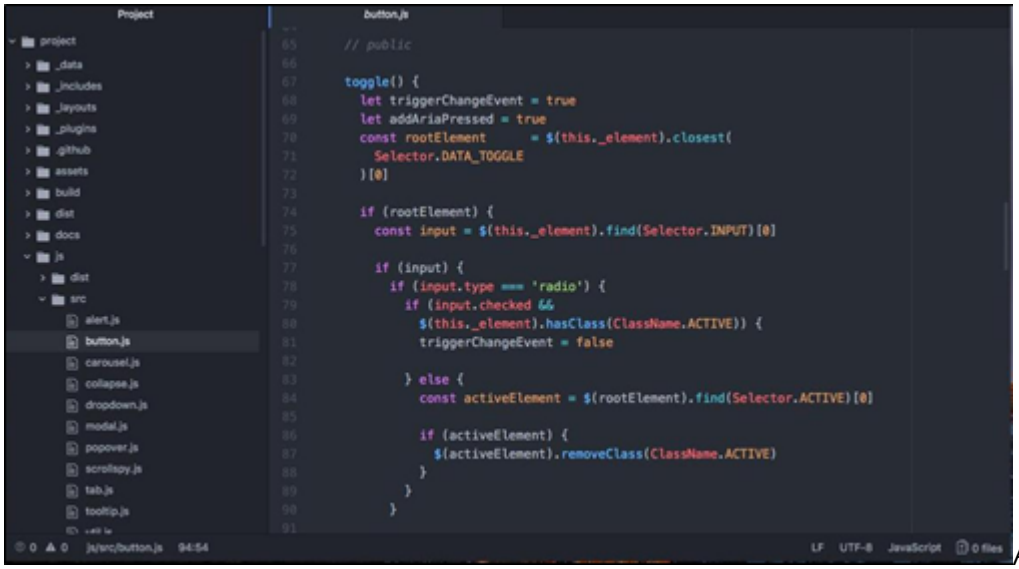
VS Code



Visual Studio Code, or VS Code, is a great IDE. It has a lot of plugins that make code readable depending on the programming language used. VS Code is used by many, for many projects. It is especially good for getting into the nitty gritty of many types of scripts and managing continuous delivery projects.

For beginners or those just exploring research, something like Spyder or Jupyter might be better than VS Code. While there are some plugins, VS Code really keeps a developer at a high-level and doesn't give the values of data or present the charts in-line quite like the others.

Atom

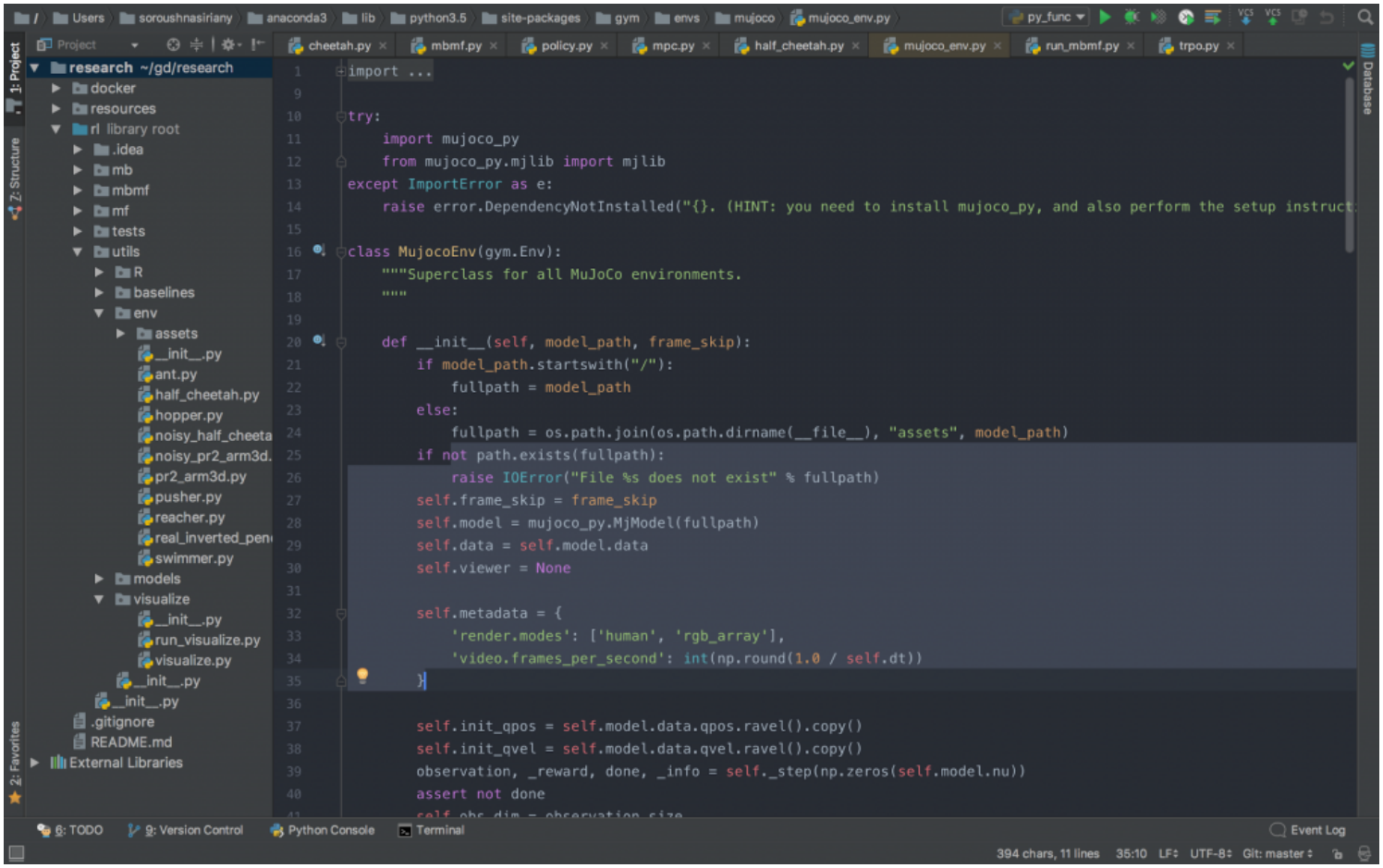


Atom is the minimalist IDE. Another popular IDE, Atom has the file manager on the left side and the code down the right. It is a good way to just dive into the code and make edits wherever they need to happen.

Atom tends to run a little slower than VS Code. But, it does have a plugin system that allow for:

- Installing themes
- Using Jupyter notebooks
- Autocompleting lines
- And more!

PyCharm

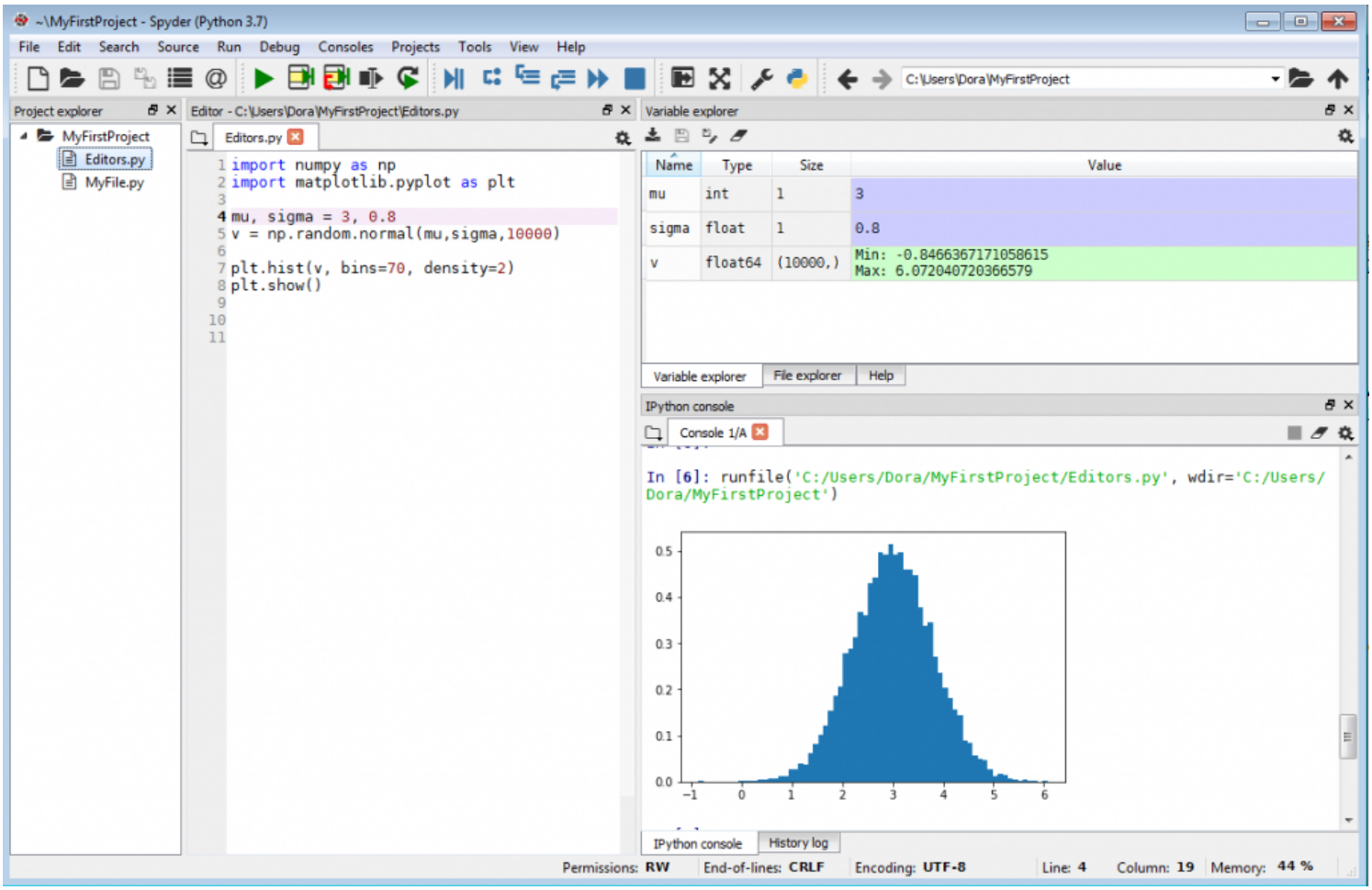


PyCharm is another good option that does it all...for Python. This editor specifically runs Python code.

It has:

- Interactive Jupyter (iPython) notebooks
- Scientific tooling (much like Spyder, below)
- Plugins to customize its coding experience.

Spyder



Spyder is great for beginners. It's even great for researchers and data scientists who need to get close to the data.

With Spyder, the code is easy to run. For example, sometimes VS Code produces indentation errors for beginners—Spyder automatically handles these. Its screen layout is clear, so you can see what files are on your system. It's also easy to know what variables you are currently working with and the values of those variables. (These variable values tend to be hidden in the other IDEs.)

Virtual Environments

The next tool in your Python toolkit are virtual environments. VEs keep projects separate. A web and server project will use different tools than a data science, graphing project. The first will use the Django and Flask packages, and the second will use Matplotlib and NumPy.

Sometimes packages can overlap, or step on each other's toes. So, instead of storing all projects' dependencies on your machine and possibly running into package conflicts, virtual environments are created. In those VEs, you can install those packages.

Another way of abstracting the dependencies a further step away from the developer's computer is

to use virtual machines, or VMs. Anaconda and Docker are tools to manage these:

- Anaconda works with virtual environments.
- Docker utilizes containers that can run on VMs.

Now, those are great steps to organize packages per project. The next item, managing your Python-specific environment, can be helped by these:

- **pyenv & pyenv-virtualenv** is for switching between versions of Python. According to its GitHub, "pyenv lets you easily switch…It's simple, unobtrusive, and follows the UNIX tradition of single-purpose tools that do one thing well."
- **Poetry** helps package systems and manage dependencies. This activities is notoriously convoluted, sometimes cumbersome, and especially tricky for beginners. "Poetry helps you declare, manage, and install dependencies of Python projects, ensuring you have the right stack everywhere"—and it does all this with a single file, the new, standardized pyproject.toml.

Python packages

Python packages and libraries grow bigger year after year. In order for companies to reach their developer users, they *invest* in building a python library. (One GitHub star might equate to $1,800 in value, for whatever that's worth.)

While you may find some niche libraries for your use case, like MedPy for medical image processing, at the very base level, there are good packages in which you should know. Here are some favorites:

Mathematics

Will you ever use math in life? These packages are for those people who discovered they need math—and forgot the basics.

Someone already coded calculus into a package, so, while you may not remember how to find the integral, nor want to write code to solve it, nor do it 1,000 times over, someone already put the basics down so you can write: Math.integral(3x). For math libraries, check out:

- NumPy
- SciPy

Graphing

For the Python project that needs to display data, these are the tools:

- MATLAB
- Seaborn
- Plotly

MATLAB is the long-standing staple, where seaborn and Plotly present those graphs with style.

Data tables

A lot of people use Python for data science, which means working with datasets: lists and .csv's. Two leading packages are:

- Pandas
- Dask

Pandas, the top choice bar none, is a great tool for examining data. When those files get too big for Pandas, go to Dask. Dask is a tool for working with files that are too large to fit in computer memory. This tooling, for datasets, is a huge component of what separates Python from other languages like Go.

Machine learning libraries

All the major ML libraries are being written in Python. The big ones are:

- scikit Learn
- TensorFlow
- Pytorch
- io

Websites and APIs

For building websites and APIs, check out:

- Flask
- Django

Build websites and APIs with your Python code using Flask. It only grows, more and more, as an acceptable way to deliver your Python backend to a consumer. Django is best suited for speedy development with clean, logical design—perhaps best used by more advanced Python programmers.

Python programming and tools

Python is the de facto programming language now and for years to come. Learning Python, especially with these tools, can be the first step in your journey to learning more programming languages. After all, once you know one, you can easily learn others.

Additional resources

For more on Python and development overall, explore these resources:

- BMC DevOps Blog
- BMC Machine Learning & Big Data Blog
- State of DevOps 2020: A Report Roundup