01_GASA_Why python_2024-05

June 20, 2025

1 License and Disclaimer

This notebook was prepared by Red Bush Analytics for the Geostatistical Association of Southern Africa. The reader can use the information below when appropriate credit is given where text is extracted. The information contained in this document is generally from public sources. The copyrights to the content contained in the links belong to the authors of the content. It is your responsibility to check any restrictions on use with the respective copyright holders.

Opinions presented in this document are those of Red Bush Analytics and Kathleen Body (Author) as of the date published (2025/06/19) and are subject to change when presented with different information. If you see another document with a different date and opinion, it is because I found new information, learned something and amended my opinion. Or, the Python code, Jupyter Notebook/lab software, or third-party links have changed, etc., and I updated the information. The information may change in the future. Check the document dates to ensure you have the information appropriate to the version of the software you are using.

The information in this document is generally appropriate for Python versions from 3.10-3.12. Substantial changes occurred from version 3.10 onwards, and the information may not be applicable to earlier versions of Python (especially, 3.8 or earlier). As of 2025/06/19, all of the links were still current. The latest version of Python is 3.13.5, a good stable version of Python is 3.12.x

2 Why start with Python?

Python is becoming like Excel, where a basic level of skill will be required to do the analytics and data manipulation portion of geological work.

The mineral resources sector is a sub-branch of Engineering, Data Science and Data Analytics. Unless you are working for a commercial software developer or writing production applications for mining operations, the most suitable languages for analytics and prediction are Python and R. SQL is used for database work.

"R" is a statistical language initially developed at Bell Labs (a research unit of the American telephone operator AT&T) from an earlier language "S". "R" is predominantly used for formal statistical work and is generally required for this type of analysis in academia. It has great visualisation capabilities and a vast support network. Python is a more general language that has a broader range of functionality than "R". Python does not have the depth of pure statistics. Still, it can do a wide range of statistical algorithms and decent graphics with a skilled programmer. Python has good data manipulation capabilities and a huge public library of solutions and packages. The original developer remains involved, and substantial development has occurred over the last few years. Python is well-suited for data science and analytics work in engineering and finance, which does not require hardcore statistics but rather a broader range of functionality. Python is more suited to users who are not professional programmers than other languages. Both Python and "R" are easy to learn, but Python is simpler and reads more like English than "R".

Modelling and estimation software from Vendors, such as Isatis.Neo, Vulcan and Micromine use Python as their scripting language. Python can also be used with Datamine to replace macros. These scripts can be used to automate processes or simply record your actions. Some software works better and faster from the scripts than from the point-and-click menus embedded in the software.

Using Python or another language like "R" allows for scripting and recording the Exploratory Data Analysis process and results. Documentation of Exploratory Data Analysis is better with scripted workflows than using miscellaneous Excel workbooks with obscure links to data "somewhere", that cannot be accessed by other users, and graphs also "some-other-where" floating in one of the worksheets in the workbook. Python has more functionality than any commercial software.

Using Python can create a single workflow between open-source and commercial modelling software. It can also add functionality instead of waiting a few years for the commercial software Vendor to add a new option. You don't have to remember the data file you used, which output file is the correct one, or where you put the files, because the Python script keeps a record for you! (Assuming you have written the script to record your selected files)

Automating boring repetitive tasks is a great stress reducer. - Scripting your repetitive work saves time and lowers the human error factor. - Set up your weekly or monthly reporting to read the new data, update the graphs and dashboards, and automatically update the PowerPoint presentation. - You can set up a logging system to record and report changes and automatically archive the old files. - Find your stuff. - Organise your stuff. - Create easy backup routines. - Set up Version control. - Reduces physical strain on fingers, arms, shoulders, and back by reducing the number of clicks.

Make very pretty pictures.

Make Peer review and audits less painful for everyone.

Learning how to write code and mini software helps you understand how software is constructed and appreciate why commercial software is so expensive. You stop complaining, " why can't they JUST write a module to ……."

3 Differences between Python and other languages.

If you are interested in learning the differences between languages, below are links to various articles that discuss the differences between the languages.

Python vs R

https://www.ibm.com/blog/python-vs-r/#The%20Main%20Difference%20Between%20R%20and%20Python:%20

IBM has a great website with a lot of interesting and valuable information.

Python vs JS or C++

https://www.linkedin.com/pulse/python-vs-javascript-which-language-should-you-learn-web-development https://www.turing.com/kb/python-vs-javascript-complete-introduction

https://www.softwaretestinghelp.com/python-vs-cpp/

Java and C https://www.geeksforgeeks.org/comparison-of-python-with-other-programming-languages/

https://medium.com/@xyuon.tech/python-vs-other-languages-which-should-you-choose-6a86031dc0d4

Compiled vs Interpreted languages

https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/

https://medium.com/@sikirus81/interpreted-vs-compiled-programming-languages-a-beginners-guide-63ed1040e973

https://www.learncpp.com/cpp-tutorial/introduction-to-programming-languages/

4 Which programming language is the best to learn?

Answer: It depends! on what you want to do. The two main divisions are interpreted and complied with. These are not so much language classes as they are ways of working. For compiled languages, the code is developed, and then the program is translated into machine language, which produces a "programme" that can be executed. Interpreters work on a line-by-line or code block-by-code-block basis and "run" the code in real time. **Compiled Languages** Traditionally, with compiled languages, the developer has to compile the code, run the code and debug if the code does not run correctly. This is time-consuming, and features must be added to the program to find where the code crashed. Development time is often long, but the program can be distributed and is usually platform-independent. Compiling is good for version control, and the program works on many systems, independent of the language. This is why the GSLIB executables still run on most computers 30+ years later. **Interpreted Languages** With an interpreter, the developer can run the code line by line or block by block, check for logical and coding errors, and debug in real-time. The code and the uncompiled code are platform-dependent. Errors in the code do not need to be referred back to the developer for correction. Interpreters now display clear error messages

indicating which line of code is causing the issue. Development time is quicker than compiled code, but requires an interpreter to run unless compiled. The user can debug uncompiled code, and the code can be used across many different interpreters. (Great for Open Source distribution, but bad for version control.) Interpreters are part of packages called Integrated Development Environments (IDEs), which programmers use for code development, debugging, and testing.

Reality is less differentiated There is less difference between the two working styles, as traditionally compiled code (C, Java) can be developed with interpreters and then compiled for deployment. Languages that generally run on interpreters, such as JavaScript and Python, can also be compiled and run as executables.

Language choice now is more about the task to be accomplished than working style.

Which language to learn? Again, it depends. Do you want to be a mobile app developer? Work in financial services to develop and run accounting systems? Data analytics, pure statistics, or developing engineering applications? Work for Microsoft and develop Office apps? Each field has its preferred set of languages depending on the tasks.

5

6 Language and Documentation.

IDEs like Jupyter allow comments (Markdown language) in multiple languages, but the Python code is in English. If your English vocabulary isn't very good, there are several courses that teach English to programmers/software developers, like https://www.freecodecamp.org/news/learn-english-for-developers-a2/. These focus on necessary vocabulary. They are also useful for English speakers who need to build a vocabulary.

Jupyter Localisation Jupyter theoretically supports localisation in terms of language in several languages. Most European languages are supported, two Chinese versions, as well as major East and Southeast Asian languages, Arabic, and Hebrew. Language packs can be downloaded and installed from https://github.com/jupyterlab/language-packs/tree/main/language-packs.

Sounds great but!

Python Documentation in non-English languages is patchy. Some parts are translated, but most of the documentation is in English. There is a search function in several languages, but the documentation is basically a mix of "other language" and English, so it is not always useful. I've tried it in French; sometimes the translation is good, while at other times it consists of a few lines of French and the rest is in English. I use a good-quality translator instead. DeepL is suitable for European Languages but is unavailable in most African countries, and some South American and Asian countries. Reverso offers a wide range of languages (mainly European and major Asian languages) and is free for short passages. Use a translator that is suitable for your language.

Help and Tutorials in other languages

While there are a number of websites offering support and tutorials in languages other than English, English is the language of computer code for the most widely used software, and most of the sites with helpful information originated in countries that speak English as a native language or strong second language, i.e. USA, India, several European countries. You can use search engines to find help and examples. As most of the information is in English, it helps to be able to search using

English. However, major search engines can search multiple languages and return useful results. You may need to set your browser to return results in multiple languages if you want to see results from more than one language. There appears to be good support in major languages with large numbers of users, such as French, Spanish, Hindi, Urdu, Russian, Korean, and Japanese, as well as some support in Indonesian and Southeast Asian languages. Additionally, there is support in Arabic and Hebrew, and limited support in African languages, such as Yoruba and Swahili. Check around for more information, as this information is subject to change frequently.

There is a nearly complete Chinese version (translation) of Python, known as Zhpy, which can be translated back into English, allowing non-Chinese readers (and computers) to read it.

7 Browsers

The Jupyter Notebook aims to support the latest versions of these browsers:

*__Chrome__

*__Safari__

*__Firefox__

I use Jupyter Notebook and JupyterLab in the latest versions of **Edge** and **Chrome**, and they appear to work without problems. There are a few challenges with **Opera**. I have not tried Safari, Firefox, other browsers, or MAC OS.

8 Computer languages used for data science and data analytics

The article in the link provides a summary of the languages used in Data Science as of early 2023 and remains valid in June 2025.

https://builtin.com/data-science/data-science-programming-languages

Python is good for Data Analytics and Machine Learning. It can be used as a scripting language in Isatis, Vulcan, and Micromine and can access Datamine (Surpac? has its own internal scripting language that needs to change, but it may be possible to access it from outside the software). There is a Python version of Isatis, a new development Deep Lime (https://deeplime.io/) and RSMP (from CCG Alberta), part of which is free and open-source. (https://www.ccgalberta.com/pygeostat_legacy/pythonintro.html)
Mines Paris – PSL(Mines Paris Tech-Fontainbleau) is rewriting most of its legacy geostats code (in C++) and has opensource packages for Python and R () https://gstlearn.org/ Michael Pyrcz-GeostatsGuy has a large library of learning materials in Python https://michaelpyrcz.com/ .

R is well-suited for data mining and statistics, and it offers excellent visualization packages for statistics. There are Python packages that allow access to "R" packages in Python scripts. In-depth geochemical analyses are a good use case for "R" in Mineral Resources. **SQL** for serious databases and as a query language for commercial database platforms like Datamine's Fusion **JavaScript** is used for scripting in Datamine, data visualization, generally, and in some commercial software **C/C++** (and legacy Fortran) are used in commercial software Many Python packages are actually written in these languages and accessed through Python wrappers. **Julia** There is some academic geostatistical code written (Julian Ortiz, RSM)

Excel-Visual Basic - Excel is not going to disappear anytime soon, and a huge amount of data is still stored in spreadsheets, some of which contain legacy code. Excel has an add-on to integrate Python functionality; however, it is a paid add-on and relatively expensive for the added functionality.

9 Integrated Development Environments (IDEs)

Software that helps developers code more efficiently has special editors and can run code or code blocks to debug code before deployment or compilation. The article on Amazon Web Services has a well-written overview.

https://aws.amazon.com/what-is/ide/#:~:text=An%20integrated%20development%20environment%20(IDE,easy to-use%20application.

The most common are: >**Jupyter Notebooks/JupyterLab/GoogleCoLab** - Based on the Jupyter Lab foundation. Widely used for Data Science, great for beginners and has good documentation. Jupyter Notebook is still supported; however, the latest version 7 does not support some extensions. The main development and newest features are found in JupyterLab. >**Visual Studio Code** is a general-purpose, rich-text editor with interpreter functions, suitable for general-purpose EDA and application development.**Microsoft Visual Studio** is a full IDE, designed for development. >**R Studio** - for R, this is almost universally used for "R" and is similar to Visual Studio in style and function. Other IDEs for specific use include PyCharm (free and paid versions) and Spyder(not for beginners).

There are others, but these are the most common in the mining and earth science communities. See the article below for others and use cases-

https://www.geeksforgeeks.org/top-10-python-ide-and-code-editors-in-2020/

10 Where to start

If you want to install Python and an IDE on your company computer, check your company's IT policy. Some companies do not allow downloading of open-source software, including Python, without prior authorisation or other permissions. They may restrict which versions you can access and the source of the software.

10.1 Installation options check the internet ofr other options

1. If you have some programming experience and can use the command prompt and PowerShell, or your commercial software does not work with Python installed in the conda environment. Download the software from the source. Follow the installation instructions in the documentation. You will need to install the required packages individually. **You need this option if you are a Micromine user.** For commercial software, refer to your user manual and follow the instructions for installing Python and other required packages.

2. If you have some programming experience and can use the command prompt and PowerShell. Choose an IDE and load the software. You can install Visual Studio Code or another IDE by downloading it from the official website. The IDE will either include the latest version of Python and major packages or provide instructions for installing them. Follow the instructions

and download Python and the required packages. You will either install Python packages via the command prompt or an IDE.

3. JuoyterLab Desktop with Python install. THUS IS THE EASIEST OPTION. Go to https://github.com/jupyterlab/jupyterlab-desktop?tab=readme-ov-file#installation and choose the correct installer for your machine. Most people will choose x64 Installer for Windows. Download the installer and install JupyterLab. When you launch Jupyter lab fr the first time you will probabaly get a message at that JL cannot find a Python environment. Choose install bundled Python.

OR

4. If you don't want the hassle of (1 or 2), have little to no programming experience, do not know what a command line prompt is, or have never used PowerShell or DOS. I suggest starting by loading the Anaconda environment https://www.anaconda.com/download/ Anaconda installer will set up VS Code, Jupyter Lab/Notebooks, and RStudio (plus other IDEs if you want them) environments and get you started. It installs the latest stable version of Python and R by default. It's a plug-and-play option. You may ned to install additional packages via your IDE.

10.2 Tutorials

5. Find some tutorials, free. Good sources of longer free tutorials are:

   - geek-for-geeks - good set of free and paid tutorials; https://www.geeksforgeeks.org/

   - Simplilearn - some free beginner courses, others are paid: Indian version of Coursera; https://www.simplilearn.com/skillup-free-online-courses
   - Coursera - some free courses. This one looks ok https://www.coursera.org/learn/python-crash-course#modules
   - Code Academy - unfortunately, the main Python course is not free, but some modules are: https://www.codecademy.com/
   - freecodecamp - has videos of lessons. https://www.freecodecamp.org/ has videos of lessons.
   - These are professionally written courses, and the first two are kept updated with the latest stable language upgrades. Code Academy and Freecodecamp versions might not be fully up-to-date, so check when they were released.

5. Find some tutorials- Good paid ones. These are primarily targeted at sales and financial examples, because that is where the real money is, not engineering and the sciences. These are professionally written courses that are kept up to date with the latest stable language upgrades (see warning for Udemy courses).

   - Geeks-for-geeks - current market leader, use the website for access.
   - Simplilearn - I have not used these, but they appear to be good.
   - Code Academy https://www.codecademy.com/learn/learn-python - there are some free skills upgrades, but the main course has a paywall
   - 365DataScience https://365datascience.com/courses/
   - 365Careers https://365careers.com/courses/ - for basic DataScience English, Chinese, Japanese, German, Spanish, and Portuguese

- Udemy—This platform offers variable quality and is targeted at sales data. Only choose the most recent (2022 onwards, using Python 3.10+) with a decent number of enrollments. The 365Careers Data Science BootCamp is excellent and is kept up to date. Jose Portilla is a popular and effective presenter, but sales-oriented. The sales-oriented courses are helpful for individuals who need to create dashboards, pivot tables, and monthly reports. Avoid any paid courses that focus on neural networks, TensorFlow, and similar topics, except for the 365 Careers Data Science BootCamp. These are challenging because some of the platforms are not very stable, have installation issues, and the courses may not be up-to-date with the latest version, among other issues, and are not for beginners.

10.3 Tools and other skills

6. Math skills https://www.coursera.org/learn/datasciencemathskills. There are more intensive options from the MIT OpenCourseWare series.

7. YouTube: As with anything on YouTube, the quality is variable, and you can waste a lot of time on junk content. I recommend sticking with structured courses, such as those mentioned above in the beginning. Once you know enough to filter out the junk content, YouTube has a lot of helpful information.

8. Stack Overflow, Quora and similar Chat forums. Just avoid them in the beginning - there is a lot of garbage here, and you will waste a lot of time on dead ends, bad programming and out-of-date code. Also, you need to know enough about Python and programming to ask the right questions using the correct vocabulary to get a useful search result. Stack Overflow provides valuable historical information, but it is often not up-to-date with the latest version of Python, so solutions may no longer be effective.

9. ChatGPT and CoPilot—again, avoid until you know enough to sift through the rubbish. ChatGPT is effective for simple problems but falls over on more complex codes. The free version has limits on queries and length, and uses an older or smaller version of the LLM. It is likely not up to date with the latest version of Python. You need one of the paid versions to get good results. ChatGPT is trained on Stack Overflow and similar sites, so it has the same problems. ChatGPT does not provide references, and plagiarism is likely to occur sooner or later; be careful. CoPilot is a live web search that retrieves more recent material and provides references. However, the code it returns is not as effective as ChatGPT and may not function as expected, requiring experience to resolve any issues. As with Stack Overflow and other chat forums, you need to have a basic understanding of Python and programming to ask the right questions, using the correct vocabulary to obtain a useful search result.

10. Learning Jupyter Notebooks/Jupyter Lab - The Documentation tab in the Notebook provides a link to information to get you started, along with a short video to get started. https://www.youtube.com/watch?v=H9Iu49E6Mxs JuoyterLab (2 years old) https://www.youtube.com/watch?v=p01wt-WB84c

[ ]:

[ ]: