INGENIERÍA GEOLÓGICA ASISTIDA

Geología desde los ojos de un dron

Una poderosa herramienta educativa para los actuales ingenieros en Ciencias de la Tierra

Proyecto DGAPA/UNAM/PAPIME: PE101020

Unidades Teóricas

UT-06
Breve panorama de las Ciencias de la Tierra y la programación en Python

Solano-Rojas Dario

UNAM / FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA EN CIENCIAS DE LA TIERRA

Este producto docente ha sido financiado por la DGAPA-UNAM a través del proyecto PAPIME PE101020 “Geología desde los ojos de un dron: una poderosa herramienta educativa para los actuales ingenieros de la Tierra”.

Por favor cítanos como:
Solano-Rojas D. (2021), Breve panorama de las Ciencias de la Tierra y la programación en Python, Unidad teórica UT-06, proyecto PAPIME clave PE101020 (DGAPA-UNAM): “Geología desde los ojos de un dron: una poderosa herramienta educativa para los actuales ingenieros de la Tierra”. Facultad de Ingeniería, UNAM.

1. Introducción

Las computadoras son cada vez más poderosas y capaces de procesar más datos. Por lo tanto, son capaces de resolver problemas cada vez más complejos, sobre todo cuando se trata de la implementación repetitiva de algoritmos y del manejo de grandes conjuntos de datos. Las Ciencias de la Tierra no están exentas del avance de la tecnología y el incremento del volumen de datos a utilizar para la exploración, los cuales tienden a ser cuantiosos. Por supuesto, una exploración geológica basada en más datos tendrá menos incertidumbre. Además, los datos geológicos tienen características geométricas y dimensionales particulares, lo cual hace complicado su análisis "a mano". En la actualidad, tanto la industria como la academia requieren de profesionales que tengan nociones básicas de programación para manejar grandes cantidades de datos en rutinas estandarizadas.

Existen muchos lenguajes de programación, todos con ventaja y desventajas. Por ejemplo, MATLAB (The MathWorks, 2015) es un lenguaje de programación muy utilizado en las ingenierías, capaz de realizar desde operaciones aritméticas simples hasta modelado y clasificaciones utilizando inteligencia artificial. Sin embargo, es una herramienta que requiere pagar una licencia, la cual no siempre está asegurada en el campo laboral. En este caso, cualquier código que se pueda tener en MATLAB, va a enfrentar la posibilidad de volverse obsoleto por la imposibilidad de ejecutarlo o reutilizarlo. Sin bien es cierto que existen otros paquetes con cierta compatibilidad con MATLAB, no todas las herramientas son soportadas. Por lo tanto, es recomendable que el lenguaje de programación a utilizar para realizar un proyecto sea de licencia abierta, lo cual permite tener acceso a todas sus funcionalidades (*spoiler alert*: vamos a usar Python).

Python es el lenguaje que marca el estándar para la industria en nuestros días. A pesar de que muchos otros lenguajes de programación tienen capacidades similares a Python, todo parece indicar que Python llegó para quedarse (aunque probablemente eso se dijo, en su tiempo, de otros lenguajes de programación). Particularmente, la comunidad geológica internacional se ha esforzado por adoptar a Python como el lenguaje estándar. El objetivo de este material, por supuesto, no es formar programadores que ganen competencias de programación a *hackatones,* si no dar un panorama general del potencial de aplicar la programación a la Ciencias de la Tierra.

2. Marco teórico de referencia

Aprender un lenguaje de programación se siente igual que aprender un idioma nuevo: al inicio, puede llegar a sentirse -mucha- frustración. Por ejemplo, la primera vez que pides una hamburguesa en un restaurante de comida rápida en Estados Unidos, esperas que te reciban con un "*good afternoon, sir*". En la práctica, es muy probable que el restaurante esté lleno, y que el encargado de la caja diga de manera directa: "*for here or to go*?". Por supuesto, ninguna clase de inglés te puede preparar para una situación tan específica. Más aún, después de aprender a pedir hamburguesas, tal vez exista la necesidad de que pidas un sándwich, pollo frito, etc., lo cual implicará nuevos retos en saber qué ingredientes ordenar, cómo pedirlos, saber si el restaurant los da gratis, la cobra extra, o tal vez si los debes de tomar tú mismo de la barra de salsas. Al inicio, enfrentar esas situaciones da pena, toma tiempo, consume energía y esfuerzo mental. Eventualmente, la práctica hace que ordenar cualquier tipo de comida se vuelva cotidiano y fácil, hasta que llegan nuevos retos para comunicarse en un ambiente diferente como el banco, a la tintorería, al súper, etc., etc. Aprender un idioma es un proceso permanente, en donde nuevos retos se agregan con el tiempo, pero con la convicción de que esa incomodidad es temporal, eventualmente se llega a dominar el léxico especial para cada situación. Lograr la fluidez en un lenguaje de programación conlleva exactamente el mismo sentimiento, con una minúscula ventaja que tiene nuestra generación: Google.

La razón por la cual Python es el lenguaje de elección para la industria hoy en día, es porque Python es gratuito, portable, poderoso y muy fácil de usar (Lutz, 2007; J. M. Perkel, 2015). Además, es aplicable para resolver problemas de la vida diaria (J. M. Perkel, 2015), y tiene ejemplos para casos muy específicos, gracias a que Python tiene una comunidad enorme, la cual desarrolla herramientas para aplicaciones específicas, de manera que no hay que empezar las cosas desde cero (Python Software Foundation, 2020).

Una de las características más importantes de la comunidad de Python, es el impulso que se le ha dado a las mujeres programadoras. Algunas de las comunidades más importantes impulsoras del movimiento de mujeres programadoras a nivel internacional son la Hackbright Academy in San Francisco, Ladies Learning Code y PyLadies. Este último grupo tiene una comunidad en crecimiento en México (En Twitter están como @MxPyladies). En

general, la comunidad de Python se ha caracterizado por procurar a grupos que tradicionalmente no se dedican a la programación.

No es coincidencia que grandes sitios webs que usamos todos los días, como Google, Youtube, Bitly, SurveyMonkey, hayan surgido a través de o adoptado Python en algún momento de su crecimiento. Incluso ArcGIS , uno de los softwares más conocidos de nuestra generación, depende fundamentalmente de Python para sus aplicaciones a gran escala (Python Software Foundation, 2020). Python tiene un sinnúmero de aplicaciones en las áreas de desarrollo web e internet, videojuegos, ciencia y cálculos numéricos, educación, negocios, desarrollo de software, entre muchos otros (Python Software Foundation, 2020).

Python viene de Monty Python, y no de la serpiente. El creador de Python, Guido van Rossum, nombró a este lenguaje de programación en 1991 en honor a un programa de comedia de la BBC, llamado *Monty Python's Flying Circus* (Lutz, 2007). La serpiente, sin embargo, es ahora la mascota que aparece en todos los libros de referencia de este lenguaje de programación. El nombre Python hace referencia tanto al lenguaje como al "intérprete". Es decir, un código está escrito en el lenguaje Python, y al momento de ejecutarlo, el intérprete de Python lo convierte en lenguaje máquina para seguir las instrucciones del código. Dicho intérprete se instala dependiendo de cuál es el "sabor" de Python que nos interesa. La instalación mínima de Python requiere un ejecutable y un conjunto de librerías ligados a él.

2.1 Lo fundamental a utilizar de Python

La comunidad científica, independientemente de su ramo, utiliza un grupo escencial de paquetes (J. M. Perkel, 2015), los cuales son: Numpy (arreglos matemáticos), Scipy (algebra lineal, ecuaciones diferenciales, procesado de señales), SymPy (matemáticas simbólicas), matplotlib (para graficado) y Pandas (análisis de datos). Cada paquete se especializa en cumplir una función específica sobre la que, a su vez, es posible construir otros paquetes (Fig. 1).

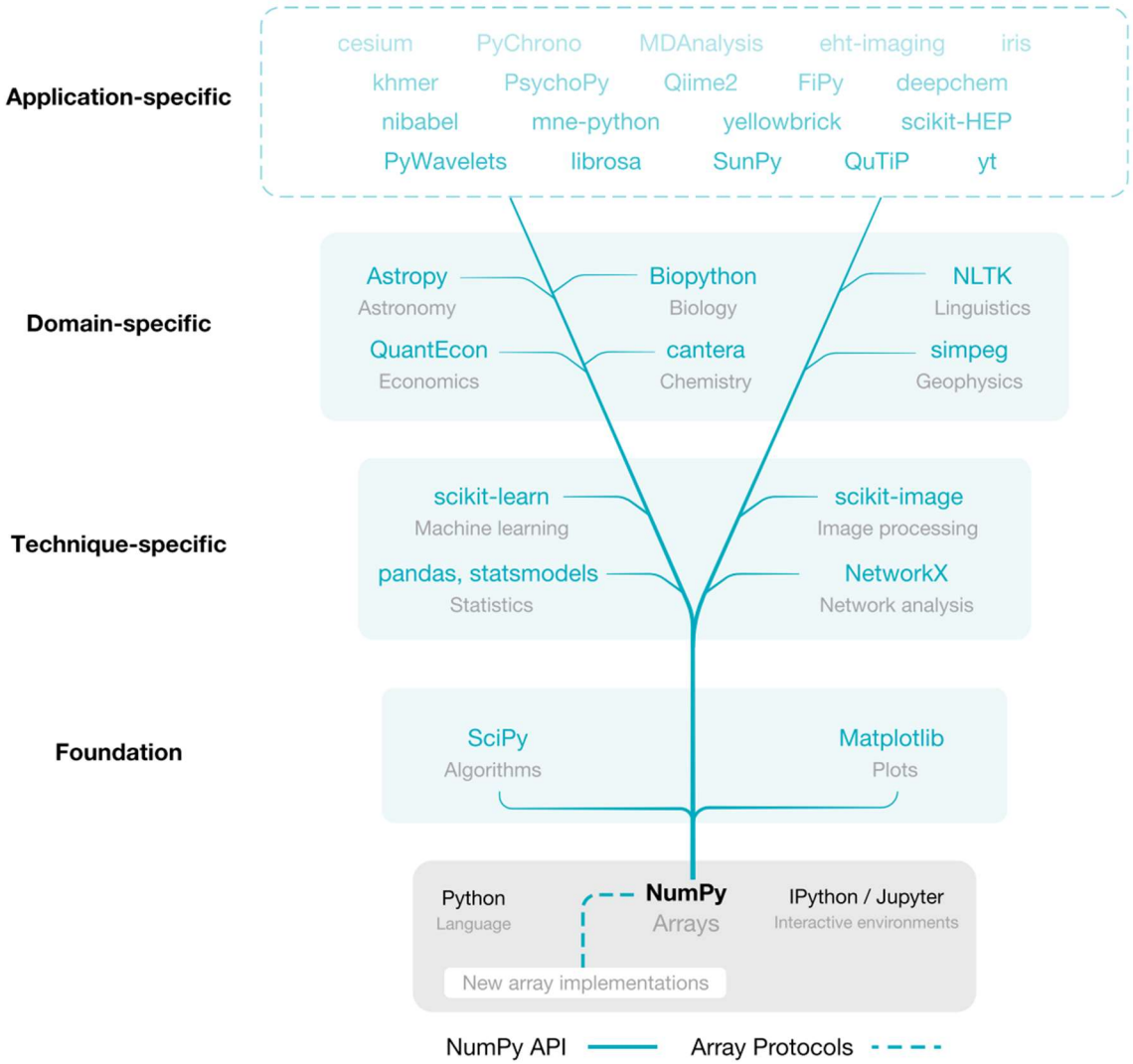

Fig. 1. Estructura del ecosistema de Python (Harris et al., 2020).

Numpy es la base sobre la que se erige el ecosistema científico para Python. Numpy es la librería de programación basada en arreglos de Python, la cual proporciona una sintaxis de compacta y expresiva para acceder, manipular y operar datos en formato de vectores, matrices, y arreglos n-dimensionales (Harris et al., 2020). Numpy es tan importante para la ciencia, que tuvo un papel fundamental para el descubrimiento de las ondas gravitacionales y la primera imagen de un hoyo negro (Harris et al., 2020). Numpy no forma parte de las librerías standard de Python. Sin embargo, el equipo que desarrolla Numpy trabaja muy de cerca con los desarrolladores de Python (Harris et al., 2020).

Scipy es una librería de rutinas numéricas para Python, la cual provee los elementos fundamentales para modelar o resolver problemas científicos (Virtanen et al., 2020). Scipy incluye algoritmos para optimización, integración, interpolación, problemas de eigenvalores, ecuaciones algebráicas, ecuaciones diferenciales, y muchos otros problemas. Scipy se contruyó sobre Numpy, por lo que tiene acceso a los arreglos y estructuras de datos de Numpy, y a su vez, permite que otras librerías se construyan encima de él. Scipy también tuvo un papel importante en el descubrimeinto de las ondas gravitacionales y en la primera imagen de un hoyo negro.

Pandas, por su parte, se enfoca a leer, manipular y preparar datos para un análisis práctico en Python. Matplotlib, es una librería para crear visualizaciones estéticas, animadas e interactivas en Python, la cual se utiliza para presentar conjuntos de datos y resultados de una manera gráfica. A pesar de que cada uno de los módulos mencionados se han convertido en el estándar de la programación en Python, cada uno es independiente de la distribución oficial de Python. Desde sus inicios, Scipy y Matplotlib se encuentran estrechamente ligados con Numpy. Juntos, y sumados a un ambiente interactivo como Jupyter y a un graficador como Matplotlib, proveen una base para la programación basada en arreglos de Python.

2.2 Notebooks interactivos para compartir código

La necesidad de compartir piezas de código de manera rápida siempre ha existido. Sin embargo, no es poco común recibir un programa y que al momento de ejecutarlo en una computadora diferente de donde fue creado, falle. Existen varias razones comunes por las que un código puede fallar al ejecutarse en una computadora diferente, entre ellas, que la versión del programa sea diferente, que las librerías necesarias no estén instaladas, o que el path no esté bien definido. También es común que después de resolver los problemas básicos de compatibilidad entre computadoras, se requiera modificar el código que nos compartieron, y es ahí en donde radica la verdadera dificultar de usar código ajeno. Sobre todo, es difícil saber si el resultado final de una rutina es correcto si no se pueden ver los productos intermedios.

La necesidad de compartir código y permitir a alguien más ejecutarlo y modificarlo, mostrando paso por paso el avance de la rutina, y además dejando notas para facilitar las cosas al nuevo usuario del código, produjo el desarrollo de una herramienta interactiva llamada IPython (I=*interactive*) (She, 2014). Posteriormente, la comunidad de usuarios y desarrolladores, quienes usualmente trabajan en más de un lenguaje de computación, desarrollaron una herramienta más poderosa que pudiera trabajar con código de lenguajes como Julia (Ju), Python (Py), y R, lo cual originó la herramienta más utilizada en nuestros días: Jupyter.

Jupyter es una herramienta web gratuita y de código abierto, conocida como un *notebook* (o cuaderno de apuntes) computacional (J. Perkel, 2018). Jupyter permite a sus usuarios combinar código, salidas de las operaciones computacionales, texto explicativo y recursos multimedia en un solo documento. Dicha herramienta es tan poderosa que se ha popularizado tanto en la comunidad científica como en la industria. La plataforma más popular para compartir código en línea, GitHub, encontró un crecimiento de *notebooks* de Jupyter compartidos por sus usuarios de solamente 200,000 *notebooks* en 2015 hasta 2.5 millones en 2018 (J. Perkel, 2018). Jupyter es una herramienta tan potente que incluso ha iniciado una tendencia nueva en la publicación de artículos científicos: que los autores incluyan un Jupyter notebook con su código en vez de copiar-pegar como texto plano.

Un Jupyter *notebook* tiene dos componentes. Primero, el usuario introduce su código en celdas a través de una página en un navegador web (i.e. Google Chrome, Mozilla, Safari). Despúes, el navegador le pasa el Código a un kernel que corre en la computadora, aunque no lo veamos, el cual regresa los resultados al navegador web para ser desplegados. El kernel típicamente vive en la computadora del usuario, auqnue también existen alternativas que pueden correr en la nube (como *Google's Colaboratory project*). Existen herramientas adicionales como JupyterLab y JupyterHub, las cuales aumentan el potencial y la usabilidad de Jupyter, pero primero lo primero. Por lo tanto, en este trabajo decidimos adoptar Jupyter notebooks para la elaboración de tutoriales o manuales interactivos para el manejo de datos de Ciencias de Tierra en Python.

2.3 Instalación de Python

Para fines prácticos, la instalación de Python que vamos a utilizar será a través de Anaconda (https://www.anaconda.com/). La razón es simple: el equipo de personas detrás de Anaconda ya sufrió, e incluso tal vez derramó algunas lágrimas, para poder asegurar la compatibilidad de Python de diferentes paquetes en diferentes sistemas operativos. Al isntalar Anaconda, se instala todo el software y librerías cercanamente relacionado a Python, de manera que, por ejemplo, se tiene acceso inmediato a Jupyter, SciPy, Numpy, Spyder, Pandas y Matplotlib.

2.4 ¿Como ejecuto código?

Existen varias maneras de ejecutar un código de Python. La más sencilla requiere utilizar el *interactive prompt* o línea de comando interactiva, la cual se puede acceder desde IPython, en la terminal, o en Spyder. Sin embargo, la gran desventaja de programar de esa manera es que todo el código se ejecuta tan pronto como se escribe, y cuando se requiere ejecutar un conjunto de acciones en un orden determinado, puede volverse caótico. Por lo tanto, usualmente resulta más conveniente hacer uso de un editor de texto, el cual servirá para guardar las líneas de código, que después se ejecutarán en el orden en el que fueron escritas. Usualmente, un código de Python tiene la terminación .py o .ipynb.

2.5 Mi primer programa en Python

El ejemplo clásico del primer programa en cualquier lenguaje de programación es imprimir en pantalla "Hola mundo". Si ya instalaste anaconda, ve a algunas de las opciones mencionadas para acceder a la línea de comando, y pon: *print("Hola mundo")* y presiona Enter o darle click en ejecutar. Listo, requisito cumplido. Nota que la sintaxis es particularmente simple comparada con algunos otros lenguajes de programación.

2.6 Reglas básicas para nombrar carpetas y archivos

Existen algunas reglas básicas de nomenclatura que se tienen que cumplir para programar. A mis alumnos, romper una de estas reglas les puede costar caro (no tanto) porque deben

traer dulces para el resto de sus compañeros de clase. Las reglas son aplicables tanto al nombre de los archivos como al *path*:

-No usar acentos ni caracteres especiales, tales como: '+$%^"?)(*
-No usar espacios, se pueden usar guiones bajos en su lugar como en mi_archivo.py
-Usar solamente minúsculas (dentro de lo posible)
-Usar la ruta más corta posible
-No comenzar los nombres de los archivos con números
-Usar puntos solamente para la extensión de los archivos
-Usar nombres descriptivos para los archivos
-Usar el sufijo v01, v02, v03, etc. para las versiones de un mismo archivo

Un programador experimentado podría darles la vuelta a estas reglas y debatir que no es necesario cumplirlas todas, ni en todos los casos. Algún otro programador que, además de experimentado, haya llegado a un nivel de iluminación superior a través de la purificación conferida por sus preciosas lágrimas derramadas a las 4 de la mañana porque su código para la entrega final no corría, podrá dar fe de que cumplir estas reglas puede ahorrarle muchos dolores de cabeza a alguien que va comenzando. Simplemente, comparemos la ruta a este script:

/Volumes/dario/geoestadistica/nube_puntos_v04.py

contra esta otra ruta:

/Volumes/Darío/My Cloud Drive/UNAM/FI/8o. semestre/Geoestadística/tareas y ejercicios/códigos de prueba/intento final?/mi código final final ahora sí finalísimo ya porfavor!.py

Es gracioso porque es real, lo he visto un sinnúmero de veces. Algunas librerías o programas no fueron hechos para lidiar con espacios, caracteres especiales, letras que solo existen en español, o con rutas complicadas o del sistema (en donde a veces Python no tiene permisos de ejecutar o editar archivos). Además, entre más complicado es el nombre de un archivo, más difícil será llegar a él y leerlo o ejecutarlo desde un script de Python.

3. Aplicaciones en las geociencias

Las carreras de Ciencias de la Tierra generalmente utilizan sets de datos de gran tamaño. Dichos sets de datos provienen logueo de núcleos de roca, información de registros geofísicos de pozo, de técnicas geofísicas en superficie, información de orientación de planos de discontinuidad, concentraciones de elementos químicos, etc. Muy frecuentemente, las tareas relacionadas a la limpieza, tratamiento y ploteo de datos son intensivas, repetitivas, y requieren seguir una serie determinada de pasos. Además, las tecnologías emergentes de apoyo para la exploración geológica (e.g. LiDAR, fotogrametría a partir de adquisiciones fotográficas con drones, percepción remota multiespectral), han abierto la puerta para la generación de grandes conjuntos de datos potencialmente aplicables a la exploración geológica. Por lo tanto, la necesidad de programar rutinas de procesado sistemático de datos se ha incrementado.

Los datos geológicos tienen particularidades que no siempre se apegan a ejemplos generalizados de programación. Por ejemplo, los datos geológicos pueden tener orientación expresada en geometría esférica, pueden pertenecer a espacios n-dimensionales con n>>3, pueden tener distribuciones multi-modales, y, aunado a las otras características mencionadas, pueden tener una coordenada tridimensional en el espacio geográfico. Por lo tanto, se cae en muchos casos especiales para el tratamiento de datos. Aprender a manejar diferentes programas para cada caso es imposible, además de que en muchas ocasiones se requiere de una licencia de paga.

Dados los muchos retos que implica el procesar y analizar datos geológicos, es necesario apoyarse de una comunidad a la que le gusten las rocas y las computadoras. Justamente, el slogan de *The Software Underground* (https://softwareunderground.org) es: *The place for scientists that like rocks and computers*. Dicha comunidad tiene canales de conversación en Slack, recopilaciones de librerías disponibles en línea, bases de datos, blogs, calendarios de eventos, e incluso mercancía. Algunas otras comunidades que se especializan en el manejo de datos geoespaciales con Python, como el Earth Lab (https://www.earthdatascience.org/tutorials/python/), Pangeo (https://pangeo.io/#what-is-pangeo) y Earthpy (http://earthpy.org). Algunos de las librerías que la comunidad de Ciencias de la Tierra ha desarrollado se mencionan en la Tabla 1. Evidentemente, existe

una cantidad de código, rutinas y librerías preexistentes que se pueden adoptar y modificar en Ciencias de la Tierra.

Tabla 1. Relación de algunas de las librerías en Python para la implementación de rutinas de procesado y análisis de datos geológicos.

	Librería	Descripción	Vínculo
Geoestadística	pyKriging	Kriging n-dinensional	https://github.com/capaulson/pyKriging
	HPGL	Librería para geoestadística de alto rendimiento	https://github.com/hpgl/hpgl
	PyGSLIB	Estimaciones de recurso mineral	https://opengeostat.github.io/pygslib/index.html
	GeostatsPy	Geostatistical Software Library reimplentada en Python	https://github.com/GeostatsGuy/GeostatsPy
	GeoStat-Framework	Simulaciones geoestadísticas	https://github.com/GeoStat-Framework
Análisis espacial	geonotebook	Jupyter n*otebooks* de la NASA para visualización y análisis geoespacial	https://github.com/OpenGeoscience/geonotebook
	Verde	Procesado de datos geoespaciales	https://github.com/fatiando/verde
Geoquímica	Reaktoro	Modelación de sistemas químicamente reactivos	https://reaktoro.org/
	GeoPyTool	Aplicación para graficado geoquímico	https://github.com/GeoPyTool/GeoPyTool
	pyrolite	Transformación geoquímica y visualización	https://github.com/morganjwilliams/pyrolite
Geología Estructural	mplStereonet	Redes estereográficas en Python basadas en Matplotlib	https://github.com/joferkington/mplstereonet
	apsg	Análisis de geología estructural avanzada y visualización basada en Matplotlib	https://github.com/ondrolexa/apsg

4. Síntesis y conclusiones

Es importante que los alumnos de las carreras de Ciencias de la Tierra se introduzcan al manejo de datos geológicos de manera programática en Python, considerando que es el lenguaje de programación que marca el estándar actual de la industria. Teniendo ese objetivo en mente, se presenta este panorama general junto con los ejercicios de este material didáctico, y así contribuir a formar profesionales con la capacidad y habilidad de procesar datos de manera programática en el desarrollo de sus actividades profesionales en la industria y la acedemia. En los ejercicios desarrollados para este trabajo se busca proveer al alumno de las herramientas básicas para agilizar la adopción, modificación y ejecución de códigos preexistentes, pero se requiere que el alumno tenga fundamentos básicos de programación. Las librerías e implementaciones deseables que los alumnos conozcan son Numpy, Scipy, Matplotlib, y Pandas, a través de una interfaz interactiva como Jupyter. La instalación básica de Python, Jupyter, y las librerías elementales mencionadas se puede lograr a través de Anaconda. Las librerías especializadas desarrolladas independientemente por una comunidad activa de amantes de las rocas y las computadoras utilizan dicha estructura básica de Python.

Si como lector has llegado al final de este documento, y decidiste darle una oportunidad a Python, te recomiendo que hagas lo siguiente:

1. Google Anaconda

2. Descarga la distribución adecuada para tu computadora y sistema operativos

3. Haz tu primer programa con *print("Hola mundo")*

Cuando logres ejecutar esos pasos con éxito, estarás listo para seguir con los ejercicios de este material. Bon voyage!

5. Referencias citadas

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array Programming with NumPy. Nature, 585(June), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Lutz, M. (2007). Learning Python. Icarus. https://doi.org/10.1016/0019-1035(89)90077-8

Perkel, J. (2018). By Jupyter, it all makes sense. Nature, 563(November), 145–146. Retrieved from https://colab.research.google.

Perkel, J. M. (2015). Pick up Python. Nature, 518(7537), 125–126. https://doi.org/10.1038/518125a

Python Software Foundation. (2020). Python org. Retrieved September 18, 2020, from https://www.python.org/

She, H. (2014). Interactive Notebooks: Sharing the Code. Nature, 515(10), 151–152.

The MathWorks, I. (2015). MATLAB Release 2015b. Natick, Massachusetts, United States.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … Vázquez-Baeza, Y. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods, 17(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Consulta también estos videotutoriales:

Manipulación y visualización de una nube de puntos en PYTHON: https://youtu.be/QvycCx8SkJs

Segmentación de nubes de puntos basada en color y K-means en PYTHON: https://youtu.be/BKFXpVIdaRI