Python para las humanidades digitales





Witton Becerra Mayorga
Adriana López
Alex L. Rojas

ECOE EDICIONES

SIL
SISTEMA DE INFORMACIÓN EN LÍNEA
COMPLEMENTO VIRTUAL

Witton Becerra Mayorga, Adriana López, Alex L. Rojas

Python para las humanidades digitales

Catalogación en la publicación - Biblioteca Nacional de Colombia

Becerra Mayorga, Witton, autor
Python para las humanidades digitales / Witton Becerra Mayorga, Adriana López, Alex L. Rojas. -- Primera edición. -- Bogotá : Ecoe Ediciones, 2023.
279 páginas. -- (Computación y tecnología de la información. Programación y desarrollo de software)

Incluye datos curriculares de los autores -- Incluye índices de bloques de código y alfabético -- Incluye referencias bibliográficas.

ISBN 978-958-503-822-6 -- 978-958-503-823-3 (e-book)

1. Python (Lenguaje de programación de computadores) 2. Humanidades - Investigaciones - Procesamiento de datos 3. Programación (Computadores electrónicos) 4. Desarrollo de software I. López, Adriana, autora II. Rojas, Alex L., autor

CDD: 005.133 ed. 23 CO-BoBN– a1131254



Área: *Computación y tecnología de la información*
Subárea: *Programación y desarrollo de software*

ECOE
EDICIONES

© Witton Becerra Mayorga
© Adriana López
© Alex L. Rojas

© Ecoe Ediciones S.A.S.
info@ecoeediciones.com
www.ecoeediciones.com
Carrera 19 # 63 C 32
Teléfono: (+57) 321 226 46 09
Bogotá, Colombia

Primera edición: Bogotá, noviembre del 2023

ISBN: 978-958-503-822-6
e-ISBN: 978-958-503-823-3

Directora editorial: Ana María Rueda G.
Coordinadora editorial: Paula Bermúdez B.
Editora de adquisiciones: Alejandra Cely R.
Corrección de estilo: Liceth Bohórquez León
Diagramación: Álex L Rojas
Carátula: Sindy Nicol Pulido C.
Impresión: Carvajal Soluciones de Comunicación S.A.S.
Carrera 69 #15-24

Prohibida la reproducción total o parcial por cualquier medio sin la autorización escrita del titular de los derechos patrimoniales.

Impreso y hecho en Colombia - Todos los derechos reservados

Tabla de contenidos

Índice de figuras **v**

Índice de tablas **ix**

Prefacio **xi**

1 Introducción **1**
1.1 Instalación 1
1.2 Conceptos básicos 6
1.3 Acceso a datos 8
1.4 Errores comunes 8

I Obtención de datos **13**

2 Tipos de datos **15**
2.1 Secuencias de caracteres 16
2.2 Falsos y verdaderos 26
2.3 Números 31
2.4 Listas 34
2.5 Tuplas 45
2.6 Diccionarios 45
2.7 Ejercicios 49

3 Trabajando con datos externos **51**
3.1 Archivos de texto sin formato 52
3.2 Archivos con formato `JSON` 58
3.3 Archivos con formato `pdf` 61
3.4 Archivos de datos en formato `csv` y la librería `pandas` . 89
3.5 Ejercicios 99

4 Trabajando con datos en la *Web* **103**
4.1 `HTML` 104
4.2 *Scraping* con `BeautifulSoup` 113

4.3 Extracción de artículos en un currículo `CvLAC` 129
4.4 Ejercicios 138

II Manipulación de datos 141

5 Manipulación de texto 143
5.1 Expresiones regulares 144
5.2 Conjuntos de datos y más de la librería `pandas` 155
5.3 Ejercicios 197

6 Procesamiento natural del lenguaje con `spaCy` 199
6.1 Instalación de `spaCy` y descarga de corpus 200
6.2 `Token`, `Span` y `Doc` 203
6.3 Reconocimiento de entidades nombradas 218
6.4 Ejercicios 227

7 Redes 229
7.1 Conceptos básicos 229
7.2 Centralidad, densidad y diámetro 240
7.3 Comunidades 246
7.4 Ejercicios 251

Referencias 253

Índice de bloques de código 255

Índice alfabético 263

Índice de figuras

1.1 Interfaz de `Jupyter Lab` 3
1.2 Interfaz de `Google Colab` 4
1.3 Navegador `Anaconda` 5
1.4 Interfaz de `VS Code 2` 6
1.5 Cuatro instancias de la clase `cubo` 7
1.6 Organización de la carpeta de trabajo 9
1.7 Palabras reservadas, las cuales fueron obtenidas con el comando `help("keywords")` 10

2.1 Indexación secuencia de caracteres 21
2.2 Visualización de la clasificación Publindex de dos revistas. El verde indica que la condición es verdadera, de lo contrario es lila 29
2.3 Ejemplos del uso de la función `range()` 34
2.4 Visualización de la comprensión de listas en el Ejemplo 2.11 38
2.5 Traducción a código Morse de una secuencia de caracteres 48
2.6 Creación de un iterable con la función `zip()` 48

3.1 Representación gráfica de la selección de caracteres de la secuencia de caracteres `NDP` 54
3.2 Creación de un objeto iterable con la función `zip()` . . . 69
3.3 Representación gráfica de las listas anidadas en el Ejemplo 3.6 74
3.4 Visualización de selección de columnas y filtración de filas en un conjunto de datos 92
3.5 Visualización de una partición de un conjunto de datos utilizando `groupby()` 97
3.6 Diagrama de dispersión del número de visitas vs. el número de descargas 98
3.7 Diagrama de caja para el promedio de descargas diarias . 98

4.1 Ilustración de una estructura básica en HTML 105

4.2 Bosquejo de la estructura de la plantilla CvLAC 114
4.3 Número de artículos por año por revista 135
4.4 Número de artículos por cantidad de autores 137

5.1 Visualización de los atributos `iloc` y `loc` en conjunto de datos 167
5.2 Distribución de revistas homologadas por categorías por año 176
5.3 Proporción de palabras en una frase para cada emoción y sentimiento con base en léxico Emolex 196

6.1 Representación gráfica de un objeto de tipo `Doc` 202
6.2 Relaciones de dependencia sintáctica para el objeto `span1` 207
6.3 Visualización de las dependencias sintácticas para cuatro frases similares 209
6.4 Representación gráfica de *tokens* como vectores 210
6.5 Proyección en dos dimensiones de los vectores de palabras más cercanos y más alejados a la palabra `'Trump'` 214
6.6 Entidades nombradas detectadas por `spaCy` 218
6.7 Entidades nombradas detectadas en un segmento del texto 'Las Campañas del Sur' 221
6.8 Pantallazo del menú para especificar el archivo de configuración del modelo de etiquetado 225
6.9 Pantallazo del archivo de configuración modificado 226
6.10 Entidades nombradas detectadas en un segmento del texto 'Las Campañas del Sur' con el modelo entrenado 227

7.1 Redes de coautoría 230
7.2 Redes de coautoría para dos artículos. El grosor de las aristas representa el número de artículos en los que los autores han colaborado 233
7.3 Red de colaboradores del profesor García Ubaque con base en su currículo CvLAC 236
7.4 Red de colaboradores del profesor García Ubaque con base en su currículo CvLAC en un posicionamiento circular . . 238
7.5 Red de colaboradores del profesor García Ubaque con base en su currículo CvLAC utilizando `pyvis` 240
7.6 Red de colaboradores del profesor García Ubaque con base en su currículo CvLAC eliminando el nodo más central . . 241

7.7 Red de coautoría de la Escuela de Matemáticas y Estadística 244
7.8 Parte de la red de coautoría de la Escuela de Matemáticas y Estadística de la UPTC con base en los currículos CvLAC 246
7.9 Comunidades de coautoría de la Escuela de Matemáticas y Estadística de la UPTC con base en los currículos CvLAC 248
7.10 Comunidades de coautoría más densas y conectadas entre ellas, de la Escuela de Matemáticas y Estadística de la UPTC con base en los currículos CvLAC 250

Índice de tablas

1.1 Librerías que utilizamos en este libro y están disponibles en la distribución `Anaconda` 4

3.1 Número de visitas y descargas de artículos de los once artículos en el volumen 45 de la revista "La Palabra" . . 91
3.2 Instrucciones básicas para filtrar filas, seleccionar columnas y calcular estadísticas en un conjunto de datos `c`, con columnas `col1`, `col2` y `col3`. La columna `col1` contiene datos numéricos y `col2` secuencias de caracteres. 93
3.3 Fecha de publicación de los artículos en la sección "Música y Literatura" 94
3.4 Estadísticas descriptivas de descargas para los artículos publicados hace menos de 5 meses 95
3.5 Sección y descargas diarias para artículos frecuentemente descargados 96
3.6 Promedio de descargas diarias por sección 96

4.1 Número de artículos publicados en la Revista de Salud Pública y Tecnura por año. 134
4.2 Número de artículos publicados por cantidad de autores en el currículo del profesor García Ubaque 136

5.1 Conjuntos de caracteres predefinidos para las expresiones regulares 145
5.2 Cuantificacores para expresiones regulares 146
5.3 Funciones del módulo `re` comúnmente usadas 151
5.4 Subconjunto de revistas con un segundo ISSN para los años 2016 y 2017 166
5.5 Instrucciones básicas para filtrar filas y seleccionar columnas con los atributos `iloc` y `loc` en un conjunto de datos `cd` 168

5.6 Filas con la homologación por año de la revista "SEL Studies in English Literature" 169
5.7 Subconjunto de revistas con ISSN igual a `1467-9442` o `0347-0520` 171
5.8 Número de artículos publicados por autor en el primer número de la Revista Tecnura 186
5.9 Número de artículos por año en la Revista Tecnura por Luis Fernando Pedraza Martínez 189
5.10 Nombre del autor con más artículos publicados, por año, en la Revista Tecnura 190
5.11 Nombre de los autores con más artículos publicados, por año, en la Revista Tecnura 191
5.12 *Top* 10 de los autores que más publican en la Revista Tecnura 192
5.13 Asignación de puntajes en emociones y sentimientos para la palabra guerra 194
5.14 Asignación de puntajes en emociones y sentimientos para la palabra loco 194

6.1 Atributos del objeto `Token` 204
6.2 Atributos para los tokens en la variable `span1` 205
6.3 Categoría gramatical y lema para los tokens en `span1`. . 206
6.4 Tipos de entidades que involucran nombres 217
6.5 Tipos de entidades que involucran números 217

Sistema de Información en Línea



Al final del libro encontrará el código para ingresar al **Sistema de información en Línea** - SIL - donde podrá encontrar todos los bloques de código incluidos en el libro.

Prefacio

Las Humanidades Digitales son un campo interdisciplinar, que combinan el estudio tradicional de las Ciencias Humanas con la programación, para abordar los fenómenos y objetos acostumbrados de estas disciplinas de una forma distinta, gracias a la programación, la cual permite abordar las investigaciones desde una perspectiva diferente.

Si bien, bajo el espectro de este concepto subyacen muchas posibilidades las que aquí se quieren expresar consisten, básicamente, en la acumulación, manipulación, análisis y estudio de datos de la literatura, la lingüística, la historia, el derecho, el periodismo etc. La base principal de estas disciplinas es, por ejemplo, los objetos físicos como libros, al ser estos digitalizados permiten su abordaje de manera distinta gracias a la recolección de datos mediante programación que permite diseñar nuevos protocolos de investigación para estudiar las humanidades de manera más amplia, con resultados más completos, y que generan novedades respecto a los tradicionales objetos de estudio.

Como antecedentes, existen varios proyectos que generaron esta nueva forma de ver el estudio de las humanidades, por ejemplo: "Michael Hart creó el Proyecto Gutenberg en 1971, que se convirtió en uno de los primeros proyectos de digitalización de libros. El objetivo del proyecto era hacer que los libros de dominio público estuvieran disponibles en línea y gratuitamente. En 2019, el Proyecto Gutenberg había digitalizado más de 60,000 libros." (Warwick, 2012, p. 13). En este mismo sentido se crea en la década siguiente el Text Encoding Initiative (TEI) que: "es un consorcio internacional que desarrolla y mantiene un estándar para la codificación de textos electrónicos. La TEI se creó en 1987 para proporcionar una metodología para la creación de textos electrónicos, y ha sido fundamental en la creación de proyectos de digitalización de textos. La TEI es ampliamente utilizada en proyectos de humanidades digitales que implican la digitalización y análisis de textos." (Warwick, 2012, p. 20). Hasta aquí de lo que trata el asunto es de la digitalización de textos.

Como se observa, la iniciativa de estos proyectos se fundamenta en la digitalización de contenido físico, especialmente libros. En este sentido, en principio, se propuso la digitalización de varios textos patrimonio de la humanidad o como móvil se tuvo en cuenta la acumulación y el rescate de ese material mediante la digitalización de estos. Así, el Proyecto Perseus: "es una iniciativa que comenzó en 1987 con el objetivo de crear una biblioteca digital de textos y recursos en áreas como la literatura, la historia y la filosofía. El proyecto también incluye herramientas de análisis y visualización de datos. La biblioteca digital del Proyecto Perseus se ha convertido en una de las más utilizadas por los investigadores de humanidades digitales." (Schreibman et al., 2008, p. 31). De la misma manera, los autores señalan los estudios de la Biblioteca de Alejandría como un pilar para esta mirada de esta nueva manera de acumular los textos: "En 1991, la UNESCO lanzó una iniciativa para recrear la Biblioteca de Alejandría en formato digital. El proyecto involucra la digitalización y análisis de textos antiguos y la creación de una biblioteca digital que se pueda acceder desde cualquier parte del mundo. Los estudios de la Biblioteca de Alejandría han sido fundamentales en la creación de una red global de bibliotecas digitales." (Schreibman et al., 2008, p. 33).

Como se observa, estos grandes referentes proponen la creación de fondos bibliográficos digitales. La pregunta es, entonces, qué pasó después de confirmar que se podían crear numerosas cantidades de textos digitales. Acá nace la perspectiva actual de las Humanidades Digitales que consiste en estudiar mediante programas por computador estos textos.

Al tener datos digitalizados en abundancia, surgen posteriormente, las metodologías para estudiar los textos. Por eso, hoy en día, la perspectiva de las Humanidades Digitales consiste en estudiar estas grandes cantidades de texto mediante programas de computador a partir de modelos de estudio más amplios y completos en palabras de Jockers:

> Digital Humanities is not just about the application of digital technologies, but also about the development of new theories and methods for humanities research, such as social network analysis, data mining, visualization, and machine learning. These approaches enable researchers to analyze large and complex datasets, uncover patterns and relationships that are not immediately apparent, and generate new insights into human culture and society." (2013, p. 34)

En conclusión, en lo que se profundiza hoy en las Humanidades Digitales

es metodologías como análisis de redes sociales, minería de datos, la visualización de datos mediante grafos que muestren los textos y sus relaciones y el ya famoso *machine learning* que ha producido una revolución como es `ChatGPT`.

Este libro está escrito para cualquier humanista que esté interesado en aprender a programar en `Python`. En nuestra experiencia, este interés usualmente viene acompañado de un problema real que el individuo quiere resolver y ha escuchado que puede ser resuelto con `Python`. Por esta razón, recomendamos que a medida que avancemos con el material, no solo se practique con los ejemplos del libro, sino con material relacionado con el problema que atañe al humanista.

En la primera parte del libro, nos enfocamos en aprender las estructuras básicas para almacenar información en `Python` y su manipulación básica. También, extraemos datos de archivos de tipo texto, archivos `pdf` y páginas en internet con la ayuda de librerías como `BeatifulSoup` y `pypdf`. Los archivos de texto y `pdf` pueden estar disponibles en nuestro disco duro o en internet, por ende, es necesario tener una conección a internet para trabajar en el material propuesto. Para lograr extraer y manipular la información en estos archivos, es fundamental conocer y manejar las estructuras básicas de `Python`, al igual que otras estructuras disponibles en librerías escritas específicamente para manipular cierta clase de información. Por esta razón, es importante tener acceso a una distribución de `Python` que tenga instaladas estas librerías por defecto. Como explicamos en el primer capítulo, emplearemos la distribución `Anaconda`, así, no nos preocupamos por instalar ninguna librería extra, ni por incompatibilidades.

A diferencia de otros libros introductorios de `Python`, donde el uso de conceptos matemáticos es el eje central para presentar todo el material, en este libro nos enfocamos en trabajar con texto. Existen varios formatos para almacenar texto, pero no los consideramos todos en este libro. Nos enfocamos en texto sin formato, y los formatos `csv`, `HTML` y `JSON`. No obstante, si es necesario interactuar con algún otro formato, creemos que con el material que aprendamos en este libro, poseemos las herramientas para leerlo y manipularlo.

Como humanistas digitales, sabemos que el texto es una fuente extremadamente rica de información. Por ejemplo, en el tiempo desde que iniciamos a leer este libro, se han creado cientos de mensajes de texto, correos, comentarios en diferentes plataformas, etc. Además, de todo el

texto que se encuentra disponible en internet y en bases de datos. Una vez que tenemos acceso a texto, debemos organizarlo de una manera estructurada y asegurarnos que no tenga errores, en la medida de lo posible.

En la segunda parte del libro, consideramos herramientas más especializadas para el manejo de texto y de conjuntos de datos. En el Capítulo 5, iniciamos profundizando en el uso de expresiones regulares para lograr generalizar patrones de caracteres. Luego, retomamos el estudio de los conjuntos de datos y aprendemos otras herramientas para su manipulación.

En los dos últimos capítulos del libro, consideramos dos herramientas comúnmente utilizadas por los humanistas digitales: grafos y procesamiento natural del lenguaje. En el Capítulo 6, hacemos una introducción al procesamiento natural del lenguaje, utilizando la librería `spaCy`. Mientras que, en el Capítulo 7, aprendemos a graficar redes y obtener estadísticas básicas de estas. Nuestro tratamiento de estos dos temas es de carácter introductorio, debido a que un tratamiento profundo requiere de conocimiento de material más avanzado.

Todos los ejemplos de este libro pueden ser replicados, ya que todas las fuentes de información son de libre acceso. Así que, invitamos a nuestros lectores a repetir y modificar nuestros ejemplos. Para terminar, este libro fue digitado utilizando `Quarto`[1], y todas las figuras fueron creadas por los autores utilizando LaTeX, `Tikz`, `mermaid` y `Python`.

[1] https://quarto.org/docs/books

1

Introducción

En este capítulo presentamos el material preliminar para aprender a programar en `Python`. Específicamente, proponemos algunas de las opciones disponibles para ejecutar los programas escritos en `Python` y una descripción breve de los términos comúnmente utilizados cuando se programa en este lenguaje. En el caso en que ya se tenga instalado `Python` en un computador o acceso a una herramienta en Internet, puede saltarse la primera sección.

Al tener un problema que podamos resolver con programación, es esencial poder ejecutar el código que escribimos. Por esta razón, es importante que tengamos acceso a `Python` o a una interfaz que nos permita escribir, probar y ejecutar nuestro código. En nuestra opinión, cuando se está aprendiendo a programar, es importante obtener una retroalimentación del código que vamos escribiendo para motivarnos a seguir adelante. Además, detectar un error de digitación o de sintaxis, en una línea de código, es mucho más fácil que cuando se tienen varias líneas. Por esta razón, en lugar de usar `Python`, directamente, aconsejamos, al interesado en este texto, que utilice un ambiente de desarrollo integrado, conocido como IDE (*Integrated Development Environment*), en lugar de la instalación básica del lenguaje. Como existen varias de estas herramientas que sirven de ambiente de trabajo, en la siguiente sección, presentamos algunas de las opciones disponibles para que se haga una elección.

1.1 Instalación

La instalación de `Python` depende del sistema operativo y la versión de este sistema. Además, la instalación básica no contiene todas las herramientas que podemos requerir en nuestros proyectos. Así que, es necesario que instalemos código extra escrito por otros programadores,

como veremos en el Capítulo 3. Por ende, lograr instalar `Python` con todas las herramientas que necesitamos en nuestros proyectos es usualmente una tarea desafiante. Existen al menos dos soluciones para este problema, las cuales presentamos a continuación para trabajar con el material de este texto.

Por un lado, podemos evitar totalmente la instalación en nuestro equipo de trabajo de `Python` y trabajar *online*. Existen varias páginas en internet que nos permiten escribir y ejecutar el código que escribimos sin necesidad de instalar nada en nuestro computador. Las aplicaciones que recomendamos tienen una interfaz similar entre ellas y la idea original proviene del proyecto `Ipython` (Pérez & Granger, 2007), que se enfoca en `Python`. Pero fue generalizada para varios lenguajes de programación con el proyecto `Jupyter`[1]. Los *notebooks* de `Jupyter` son una de las maneras más populares de trabajar en `Python` y compartir código con otras personas. Al momento de escritura de este libro, la IDE con mayor flexibilidad del proyecto `Jupyter` es `JupyterLab`[2]. En la Figura 1.1 presentamos un pantallazo de esta interfaz. Como se puede observar en esta figura, el *notebook* contiene celdas en las que se puede digitar código directamente y, si no existen errores, se puede ejecutar.

Con `JupyterLab`, ya podríamos escribir nuestra primera línea de código; sin embargo, esta aplicación tiene solo el objetivo de que se pruebe cómo funciona y luego se descargue y sea utilizada localmente en nuestro equipo.

Otra opción disponible en internet es `Google Colaboratory`. Como su nombre lo indica, es una aplicación desarrollada por `Google` con un ambiente como el de los *notebooks* `Jupyter` que se ejecuta en la nube; luego, no requiere instalar nada en nuestro equipo de trabajo, pero requiere que tengamos una cuenta de `Google`.

En la Figura 1.2, podemos observar que la interfaz de `Google Colaboratory` es similar a la interfaz de `JupyterLab`. Para accesar a dicha interfaz, debemos ingresar la información de una cuenta de `Google`, y nuestros programas quedarán almacenados en `Google Drive` en una carpeta llamada `Colab notebooks`. Un problema que puede surgir al utilizar `Colab` junto con el material de este libro es la disponibilidad dentro de `Colab` de los datos que utilizamos. Por lo tanto, en algunos

[1] https://jupyter.org/
[2] https://jupyter.org/try-jupyter/lab/

casos, será necesario cargar los datos manualmente para poder tener acceso a ellos.

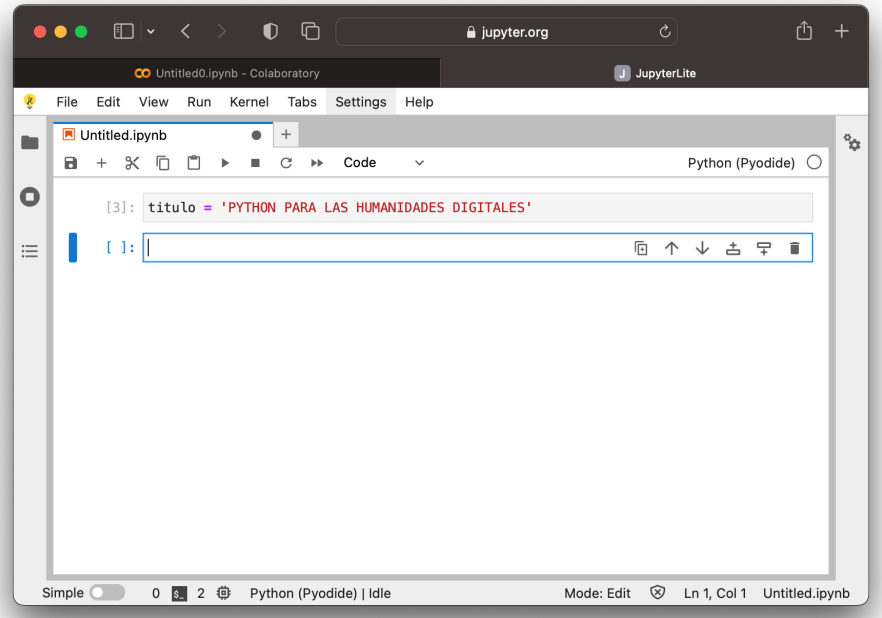


Figura 1.1: Interfaz de `Jupyter Lab`

La última opción que consideraremos es instalar `Python` localmente en nuestro equipo de trabajo. Esta tarea puede que sea complicada, por lo tanto, nuestra recomendación es utilizar un ambiente que instale y tome control sobre esta instalación. El ambiente más popular, y nuestra recomendación, es `Anaconda`. Este ambiente puede ser descargado de la página https://www.anaconda.com/download. Una vez en la página, está detectará el sistema operativo en el que se está trabajando, y al digitar `Download`, se iniciará la descarga. Para instalar `Anaconda`, debe seguirse las indicaciones del instalador.

En breves palabras, `Anaconda` remueve la necesidad de aprender cómo instalar varias de las herramientas más populares para programar. En particular, en la Tabla 1.1 presentamos una lista de las librerías que vamos a utilizar en este libro y que se encuentran dentro de esta distribución. Una librería es una colección de código reusable que nos permite realizar tareas específicas.

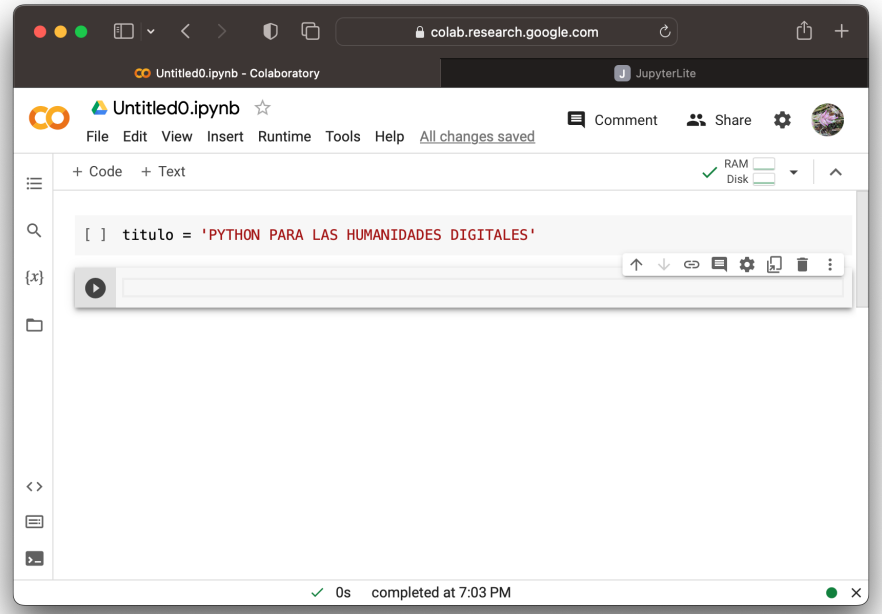


Figura 1.2: Interfaz de `Google Colab`

Tabla 1.1: Librerías que utilizamos en este libro y están disponibles en la distribución `Anaconda`

Librería	Descripción breve
`BeatifulSoup`	Crea una estructura que permite tener acceso a texto y otra información disponible en archivos con formato `html`
`matplotlib`	Es la librería más popular para crear y manipular gráficas
`pypdf`	Lectura y manipulación de archivos `pdf`
`pandas`	Lectura, manipulación y escritura de conjuntos de datos

`Python` divide su código en librerías, paquetes y módulos. Las librerías son una colección de paquetes, las paquetes reunen varios módulos, y los módulos es una colección de código que define variables, funciones y clases. La idea detrás de esta organización es dividir tareas en subtareas, de tal

manera que el código pueda ser revisado y modificado más facilmente. En la práctica, sin embargo, existen algunos paquetes que son tan complejos y tienen varias aplicaciones que se les llama librerías. Por ejemplo, el paquete `pandas`, al que le llamamos librería en la Tabla 1.1. Por esta razón, es común en la práctica intercambiar estos dos términos.

Debemos tener en cuenta que, aunque ya esté instalado este ambiente, aún necesitamos seleccionar cómo interactuamos con `Python`. Cuando se abre el navegador de `Anaconda`, el cual puede tomar varios segundos, se observará una pantalla como en la Figura 1.3. En este navegador podemos seleccionar el IDE que queremos utilizar para escribir nuestro código. Las tres opciones más populares son `JupyterLab`, `Spyder` y `PyCharm`. Al ejecutar `JupyterLab`, se abre una ventana en un navegador de internet y, como mencionamos anteriormente, podemos ejecutar código línea por línea, o mejor, celda por celda. Por otro lado, `Spyder` y `PyCharm` son comúnmente empleados para ejecutar todo un programa, no línea por línea de código. Nuestro consejo es iniciar con `JupyterLab` y, cuando hayamos ganado experiencia, cambiar, si se quiere, a emplear `Spyder` o `PyCharm`.

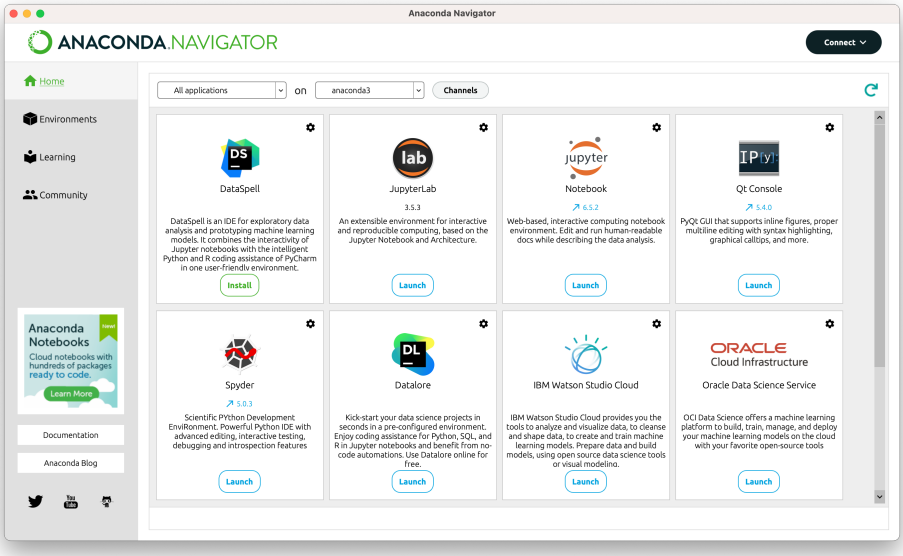


Figura 1.3: Navegador `Anaconda`

Para terminar esta sección, existe otra IDE que no está disponible con `Anaconda`, pero que también es popular entre programadores: `VS Code 2`.

Como podemos observar en la Figura 1.4, `VS Code 2` tiene una interfaz similar a los *notebooks* de `Jupyter`; pero, a diferencia de `JupyterLab`, `VS Code 2` no es ejecutada en un navegador, sino que es una aplicación por sí sola que ofrece otras herramientas.

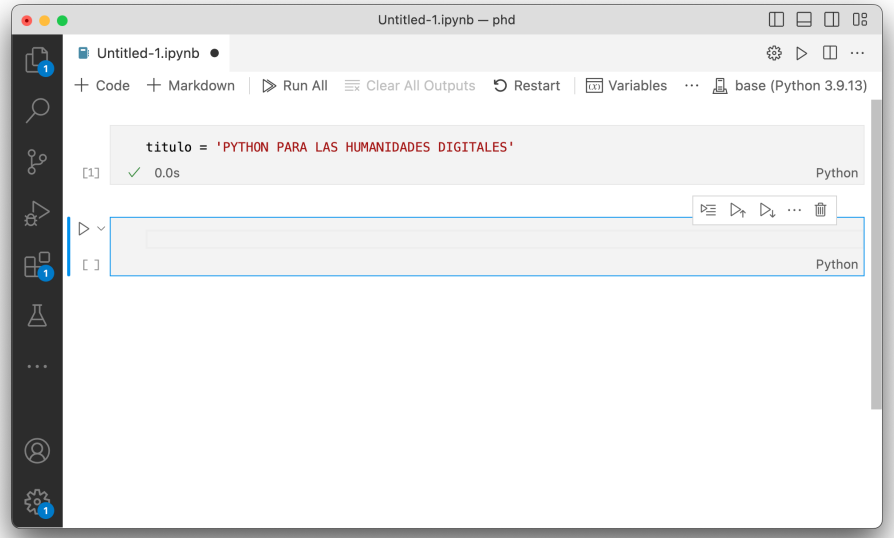


Figura 1.4: Interfaz de `VS Code 2`

1.2 Conceptos básicos

`Python` es un lenguaje de programación con múltiples paradigmas, que, en particular, soporta programación orientada a objetos. Para esto, `Python` permite definir *clases* de objetos. Con cada clase, se modela un tipo de objeto en particular, especificando los datos que contiene cada objeto y el comportamiento de estos datos. En otras palabras, una clase funciona como una plantilla a la que podemos asignarle valores específicos, generando así una instancia de la clase a lo que llamamos objeto. Un ejemplo clásico de estos conceptos que permite entenderlos mejor, es un conjunto residencial en el que el plano de las casas corresponde a la clase y las casas construidas corresponden a los objetos. Presentamos otro ejemplo a continuación.

Ejemplo 1.1 (Cubos)**.** Supongamos que tenemos una *clase* llamada `cubo`. Esta clase crea una imagen en nuestro cerebro de un cubo y es posible decidir, por ejemplo, el color y el tamaño de este cubo. Así, este concepto de clase es abstracto. No obstante, al graficar cuatro objetos de tipo `cubo` a los cuales se les asignó colores y tamaño, como en la Figura 1.5, estos *objetos* dejan de ser abstractos ya que los podemos visualizar de acuerdo a sus atributos, es decir, los colores y tamaños asignados.

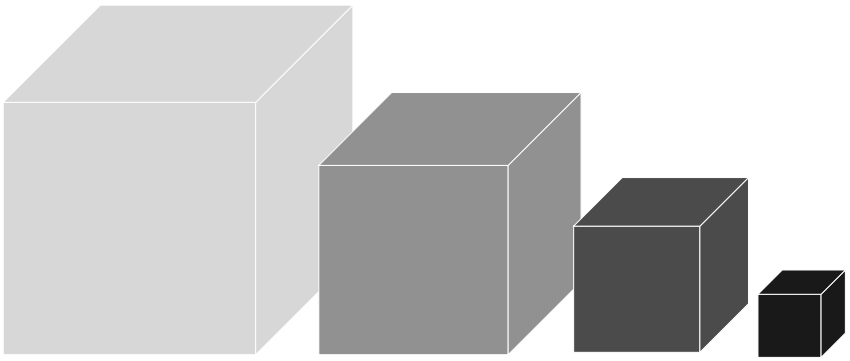


Figura 1.5: Cuatro instancias de la clase `cubo`

En este libro no profundizamos en la creación de clases, ni ahondamos en el concepto de programación orientada a objetos, pero varias veces mencionaremos el concepto de *objeto*, luego es importante tener una idea básica de qué significa. Por ejemplo, nuestra interacción con `Python` usualmente requiere que creemos datos, o que se importen datos disponibles en internet, o en cualquier otro lugar al que tengamos acceso. Estos datos deben ser almacenados en algún lugar en la memoria del computador y deben tener un nombre con el cual podamos tener acceso directo a estos. El nombre con el que obtenemos acceso a los datos lo conocemos como *variable*. Por otro lado, los datos que almacenamos tienen diferentes características. Por ejemplo, podemos almacenar el texto de un libro o una imagen, los cuales son objetos diferentes, así que el formato de almacenaje también es diferente. En otras palabras, las características de cada uno de los objetos afectan en cómo los datos son almacenados en la memoria.

La mayoría de código que presentamos en este libro lo organizamos en

"Bloques de código" que referenciamos en el texto y, además, una lista de todos estos se encuentra en el índice de bloques de código al final del libro. En estos bloques, por lo general, no incluimos comentarios debido a restricciones de espacio. Por otro lado, todo el código utiliza un tipo de letra monoespaciado; de esta manera, se diferencia el texto del libro del código. Además, siempre que hablemos de caracteres y texto dentro de `Python`, utilizaremos comillas simples ,`''`, con letra monoespacio. Finalmente, también escribimos con letra monoespacio los nombres de lenguajes de programación, nombres de archivos, librerías, atributos, *funciones* y *métodos*. En el caso de las funciones y métodos, siempre incluimos paréntesis para diferenciarlos.

1.3 Acceso a datos

En este libro trabajaremos con datos disponibles en internet de libre acceso. Por lo tanto, todos los códigos presentados y ejercicios propuestos podrán ser realizados sin problemas de acceso. Por lo general, estos datos no necesitan ser almacenados en el disco duro de nuestro computador. Sin embargo, es muy probable que en nuestro trabajo con datos propios debamos almacenar la información localmente. Nuestro consejo es que cada proyecto tenga su propia carpeta y quizá esta carpeta tenga subcarpetas como `Datos`, `Codigo` y `Figuras`. Siguiendo esta propuesta, para este libro, vamos a suponer que la carpeta de trabajo es `phd`, todo nuestro código está almacenado en la carpeta `Codigo`, los datos, ya sean documentos de texto o imágenes, están almacenados en `Datos` y las figuras que generemos serán almacenadas en `Figuras`. En la Figura 1.6 presentamos un esquema de la organización de las carpetas. Con el objetivo de mostrar la estrutura, hemos incluido algunos de los archivos con los que vamos a trabajar en próximos capítulos, y no es necesario, por ahora, que estén en nuestro directorio de trabajo.

1.4 Errores comunes

Una de las características de `Python` es que cuando ocurre un error de ejecución en el código se imprime en pantalla una descripción extensa

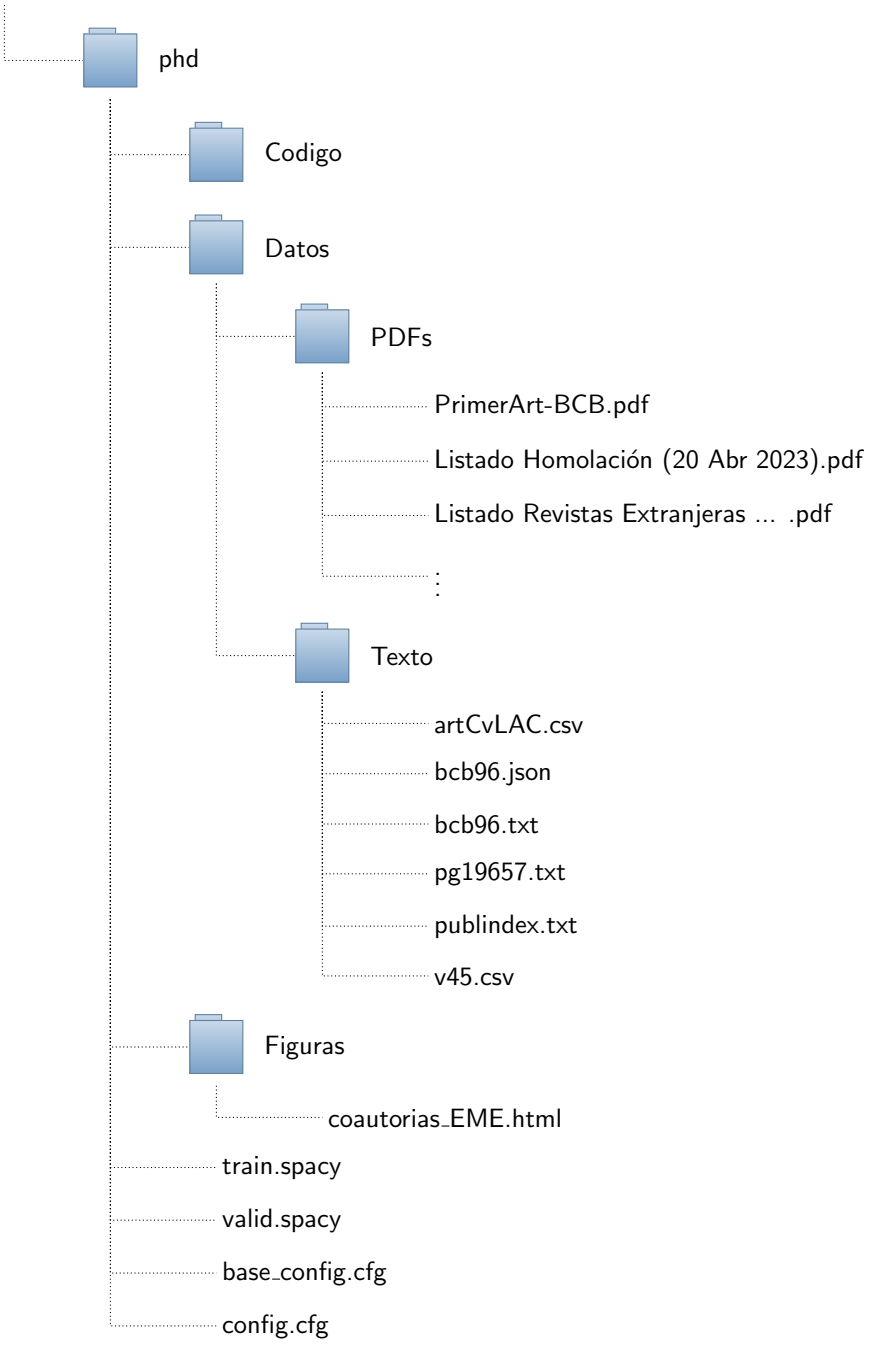


Figura 1.6: Organización de la carpeta de trabajo

del error. El problema de esta descripción para programadores novatos es que, en nuestra opinión, es intimidante y puede causar frustración. Nuestro consejo frente a esta situación es revisar, inicialmente, solo la última línea del mensaje de error. Es en esta línea donde, usualmente, encontramos la información que nos permite corregir el error. Ahora bien, y aunque suene trivial, o quizá pretencioso, la mejor manera de manipular errores es no cometerlos. Esto es obviamente una tarea imposible, siempre habrán errores, pero existen dos acciones fáciles que podemos tomar para prevenir un porcentaje alto de errores. Estas son: digitar correctamente el nombre de las variables o *funciones* y revisar que los datos o el objeto al que estamos accesando tenga la dirección correcta. Varias lectoras y lectores puede que crean que esto poco les pasará, pero, en la experiencia se corrobora, siempre nos pasará, así seamos cuidadosos.

Una de las características de `Python`, como casi todo lenguaje de programación, es que los nombres son sensibles a las mayúsculas y minúsculas. Es decir, todas las siguientes opciones son diferentes: `Autor`, `AUTOR`, `autor`, `autoR`, etc. Por esta razón, debemos siempre revisar que hemos digitado todos los nombres que usemos correctamente. Por otro lado, el lenguaje tiene 35 o 36 palabras, dependiendo de la versión de `Python`, que son reservadas y no pueden ser usadas como nombres por nosotros. Estas palabras pueden ser obtenidas al digitar y ejecutar el comando `help("keywords")`, como se presenta en la Figura Figura 1.7.

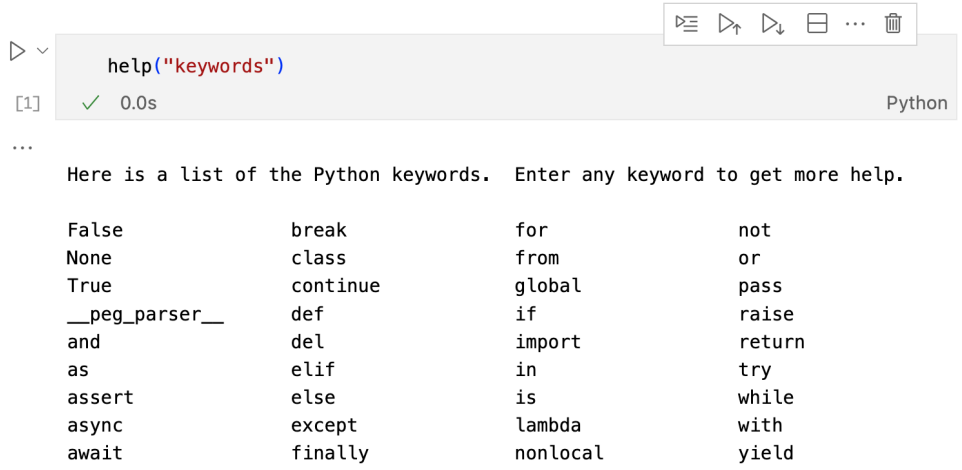


Figura 1.7: Palabras reservadas, las cuales fueron obtenidas con el comando `help("keywords")`