UNIVERSIDAD DE CÓRDOBA

Facultad de Ciencias

Grado en Bioquímica

TRABAJO FIN DE GRADO

Desarrollo de un tutorial de programación en Python orientado al estudiante de Bioquímica

---

Código del TFG: BQ-21-38-IAN

Tipología: Trabajo docente

Autor: **Víctor Jesús FERNÁNDEZ RAMÍREZ**

6 de junio de 2022

Índice general

Índice de figuras

Resumen del TFG

Resumen

El estudiante del Grado de Bioquímica en la Universidad de Córdoba adquiere una gran cantidad de competencias, entre las que se incluyen competencias bioinformáticas. Entre estas, destacan los conceptos aprendidos sobre el empleo del sistema operativo *Linux*, y el uso del lenguaje de programación *R*.

En este Trabajo de Fin de Grado se presenta un tutorial aplicado al estudiante de Bioquímica sobre un lenguaje de programación denominado *Python*. *Python* se trata de uno de los lenguajes más utilizados en la actualidad por empresas de todo el mundo, destacando por su potencia y facilidad de aprendizaje. Debido a esto, en el mundo de la Biología Computacional se ha incrementado su uso, siendo una competencia directa en el uso de *R* y su librería *Bioconductor*, gracias a la aparición de librerías como *Biopython*.

A lo largo del tutorial el estudiante aprenderá *Python* desde cero, partiendo desde su descarga e instalación, hasta llegar a conceptos más avanzados como el uso de estructuras de control, diseño de gráficas y uso de librerías específicas. Para ello se le irán proponiendo una serie de ejercicios complementarios, para que pueda ir poniendo en práctica los conocimientos adquiridos.

Sumado a esto, el estudiante también aprenderá conceptos básicos sobre una serie de herramientas muy empleadas en el campo de la Informática, como son *Markdown, MkDocs* y *GitHub*.

Palabras clave: Tutorial, Python, PyCharm, Markdown, Bioquímica, Bioinformática.

Abstract

The student of the Degree in Biochemistry at the University of Córdoba acquires a large number of skills, including bioinformatics skills. Among these, the concepts learned about the use of the operating system *Linux*, and the use of the programming language *R* stand out.

This Final Degree Project presents a tutorial applied to the Biochemistry student on a programming language called *Python*. *Python* is one of the languages most used today by companies around the world, standing out for its power and ease of learning. Due to this, its use has increased in the world of Computational Biology, being a direct competition in the use of *R* and its library *Bioconductor*, thanks to the appearance of libraries such as *Biopython* .

Throughout the tutorial the student will learn *Python* from scratch, starting from its download and installation, to more advanced concepts such as the use of control structures, graphic design and the use of specific libraries. For this, a series of complementary exercises will be proposed, so that you can put into practice the knowledge acquired.

In addition to this, the student will also learn basic concepts about a series of tools widely used in the field of Bioinformatics, such as *Markdown, MkDocs* and *GitHub*.

Keywords: Tutorial, Python, PyCharm , Markdown, Biochemistry, Bioinformatics.

Capítulo 1

Contexto docente

Este tutorial de programación en Python orientado al estudiante de Bioquímica se enmarca dentro del ámbito de la materia *Informática aplicada a la Bioquímica*. Esta asignatura se imparte durante el primer cuatrimestre del segundo curso del Grado de Bioquímica de la Universidad de Córdoba [13]. En esta materia se familiariza al alumnado de Bioquímica con un conjunto de herramientas informáticas que le pueden ser de utilidad en su carrera profesional. Por ello, abarca temas como el funcionamiento básico de un ordenador, manipulación de base de datos científicas, programación computacional en *R* y su entorno *Rstudio* para optimizar el tratamiento de datos biológicos, manejo de *Linux* y edición de ficheros en LaTeX. No obstante, aunque esta ha sido la principal asignatura en la que se han aprendido estos aspectos bioinformáticos, presenta competencias cruzadas con otras asignaturas del Grado de Bioquímica como *Biología Molecular de Sistemas*, *Biofísica* y *Estructura de Macromoléculas.*

El Grado de Bioquímica en la Universidad de Córdoba se basa en el estudio de los constituyentes químicos de los seres vivos y sus transformaciones, así como de las estructuras e interacciones de las macromoléculas que permiten la vida de estos organismos. Este concepto deriva cada vez más hacia el término de Bioquímica y Biología Molecular, ya que la complejidad de estructuras, organizaciones y funciones de los seres vivos a nivel molecular es fundamental en la comprensión de la vida [7]. La investigación tanto básica como aplicada en Bioquímica y Biología Molecular ha avanzado de una forma muy significativa en el siglo XX, continuando su avance exponencial en el siglo XXI.

Las Biociencias Moleculares y sus aplicaciones están consideradas una de las grandes olas de expansión de la economía basada en el conocimiento. De hecho, la Unión Europea ha apostado claramente por convertirse en una de las principales líderes mundiales en la economía basada en el conocimiento. La generación de conocimiento es fundamental en el abordaje de necesidades globales relacionadas con la salud, alimentación, medio ambiente y sostenibilidad. *La Agenda 2030 para el Desarrollo Sostenible* plantea un marco ambicioso de objetivos y metas universales e indivisibles para hacer frente a una variedad de desafíos societarios mundiales. La

diversidad biológica y los ecosistemas figuran en forma destacada en muchos de los *Objetivos de Desarrollo Sostenible (ODS)* y metas asociadas, en las que la Bioquímica y Biología Molecular tiene un papel crítico [21].

La importancia de la Bioquímica y Biología Molecular también ha dejado huella a nivel local en Córdoba, donde encontramos el Parque Tecnológico Rabanales XXI con industrias biotecnológicas y biomédicas, y el Instituto de Biomedicina Maimónides de la Universidad de Córdoba que incorpora profesionales bioquímicos en investigación biomédica avanzada. En el ámbito agroalimentario resalta el Campus de Excelencia Agroalimentario, liderado por la Universidad de Córdoba y que agrupa a las Universidades de Cádiz, Huelva, Jaén, y Almería, y que requiere para su desarrollo graduados formados en estas tecnologías agroalimentarias, y muchas de las destrezas a nivel molecular que posee el graduado de Bioquímica.

Capítulo 2

Justificación

La Bioinformática es una subdisciplina de la Biología y de la Informática que se ocupa de la adquisición, almacenamiento, análisis y difusión de datos biológicos. En la actualidad, la parte más importante de la Bioinformática es el análisis e interpretación de los datos de las moléculas biológicas, un proceso denominado Biología Computacional [15]. Por tanto, la Bioinformática es importante por su gran aporte al desarrollo de la Biología en la investigación básica, haciendo posible el tratamiento de la extensa información que se puede obtener a partir del estudio de organismos: genomas, proteomas, transcriptomas y metabolomas.

En base a la importancia actual y futura de la Bioinformática, la impartición de la asignatura *Informática aplicada a la Bioquímica* es fundamental en el Grado de Bioquímica. A lo largo del Grado hay otras asignaturas en las que se tratan herramientas bioinformáticas, destacando especialmente *Biología Molecular de Sistemas* y *Estructura de Macromoléculas*. No obstante, teniendo en cuenta el total de 38 asignaturas impartidas, el porcentaje de competencias bioinformáticas es relativamente bajo.

En la asignatura *Informática aplicada a la Bioquímica* el alumnado de Bioquímica adquiere conocimientos básicos acerca del lenguaje informático *R* [13]. Este lenguaje, uno de los más utilizados en Bioinformática, no deja de ser una buena opción para ser utilizado por un bioquímico. Sin embargo, no destaca por su simpleza, y si el bioquímico conoce únicamente R puede verse limitado a la hora de utilizar otra serie de herramientas, programas o librerías.

Por este motivo en el presente trabajo se propone un tutorial de *Python* orientado al estudiante de bioquímica. Uno de los objetivos por los que fue concebido *Python* en sus principios fue para ser un lenguaje fácil de aprender. La elegante sintaxis de *Python* y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para el desarrollo de programas y aplicaciones en diversas áreas y sobre la mayoría de las plataformas. De hecho, para confirmar el peso de *Python* a día de hoy, puede comprobar que grandes gigantes como Instagram, Google y Spotify lo eligen para el backend de sus páginas web [24].

Esta tendencia al uso de *Python* también ha afectado al campo de las Biocien-

cias Moleculares. Esto se ha debido principalmente al incremento en disponibilidad de librerías de visualización y procesamiento de datos como *NumPy, SciPy, Pandas* o *Matplotlib*. Este incremento ha conllevado a que actualmente un bioquímico pueda realizar las mismas funcionalidades en *Python* que con *R*, sumado a la aparición de librerías específicas como *Biopython* que se asemejan a *Bioconductor* de *R* [3]. Aun así, hay que tener en cuenta que ambas librerías al igual que contienen programas en común, no compartirán muchos otros. Este es el factor diferencial que debe tener en cuenta: es fundamental que adquiera competencias cruzadas, ya que en situaciones problemáticas en las que un lenguaje no le pueda aportar una solución, otro quizás sí lo haga.

Además, ha de saber que el aprendizaje del lenguaje de programación *Python* le abrirá como bioquímico muchas puertas en su futuro como profesional, ya que cada vez son más los laboratorios de investigación que requieren de bioinformáticos para tareas como análisis de archivos de secuenciación, modelado de proteínas y alineamiento de secuencias en este lenguaje de programación [5].

Capítulo 3

Objetivos

El objetivo principal del presente Trabajo de Fin de Grado es el desarrollo de un tutorial de *Python* desde cero, para que un estudiante de Bioquímica sin ningún conocimiento en este lenguaje de programación adquiera los sufientes conceptos básicos para manejarse en el mismo. De hecho, este tutorial puede ser utilizado por alumnado externo al Grado de Bioquímica, presentando la única limitación de que la mayoría de ejemplos y ejercicios propuestos están enfocados en el campo de la Bioquímica y Biología Molecular.

El desarrollo del tutorial está orientado de tal forma que el alumnado de Bioquímica no va a adquirir conocimientos de programación que le puedan ser tediosos y extensos. En todo momento se ha tenido en cuenta en su diseño que un alumno de Bioquímica no es un informático, y no se puede pretender que lo sea, ya que esta tarea le resultará muy compleja en la mayoría de las ocasiones. Por este motivo, el tutorial está dividido en una serie de apartados o secciones, partiendo de conceptos muy básicos y terminando con ejercicios con mayor complejidad, todo desde un punto de vista muy práctico que le supondrá una vía fácil de aprendizaje.

Sumado a esto, una de las finalidades de este trabajo es que el alumnado siga adquiriendo y desarrollando una serie de competencias definidas en el Grado de Bioquímica: capacidad de razonamiento crítico, aplicación de los principios básicos del método científico, capacidad de aprendizaje autónomo, y empleo de herramientas básicas para la comunicación, búsqueda de información y tratamiento de datos [8].

Asimismo, todo el material desarrollado en el tutorial, desde el código fuente del servidor web hasta los scripts de *Python*, están incluidos en el siguiente repositorio de GitHub: `https://github.com/Victorffdez/Python_Biochemistry_Tutorial`

En este repositorio, el contenido está accesible a toda la comunidad educativa de forma totalmente gratuita, buscando llegar al máximo porcentaje posible del sector educativo externo a la Universidad de Córdoba.

Capítulo 4

Desarrollo del trabajo docente

4.1. Tecnologías usadas en la realización del tutorial

4.1.1. PyCharm

Para escribir, leer y editar código de la manera más eficiente posible, se necesita un editor de texto o un entorno de desarrollo integrado *(IDE, Integrated development environment)*. Un IDE no deja de ser un editor de texto, pero que incluye otra serie de herramientas que le permiten examinar el código mientras se introduce y aprender sobre el mismo [17].

PyCharm es un IDE muy popular desarrollado por la compañía *Jetbrains*, construido específicamente para programar en el lenguaje Python. La versión gratuita se denomina *PyCharm Community Edition*, y contiene una gran cantidad de herramientas y opciones disponibles [18].

En este tutorial, PyCharm ha sido el IDE seleccionado para escribir, leer y editar scripts de Python; así como para escribir los ficheros Markdown que componen las distintas secciones del tutorial. De hecho, en la sección 4.2.3 del tutorial se trata exclusivamente la manipulación de PyCharm. Aunque este IDE haya sido el seleccionado, existen muchos otros IDE y editores de texto empleados por la comunidad científica, destacando Sublime Text y Visual Studio Code (VSC).

4.1.2. GitHub

GitHub es un servicio basado en la nube que aloja un sistema de control de versiones (VCS) llamado Git, siendo por tanto, un servicio de alojamiento en la Web de repositorios remotos Git [17].

GitHub permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso. Por estos motivos, GitHub es una herramienta muy utilizada por la comunidad bioinformática, existiendo miles de repositorios disponibles de programas que pueden resultar de

utilidad, totalmente gratuitos.

GitHub ha sido empleado en el desarrollo de este tutorial con dos objetivos: incluir todo el material desarrollado en un repositorio accesible por la comunidad estudiantil, y facilitar la comunicación y *feed-back* entre el autor y el tutor del trabajo en todo momento.

PyCharm es un IDE totalmente adaptado al uso de GitHub. Como aparece en la sección 4.2.3 del tutorial, la interfaz de PyCharm permite administrar proyectos de Git alojados en GitHub de forma muy simple, pudiendo crear un repositorio propio en la opción *Create git repository*, compartir el proyecto en GitHub en la opción *Share project on GitHub*, sincronizar los archivos desde el ordenador local al repositorio con la opción *Push*, y sincronizarlos desde el repositorio al ordenador local con la opción *Pull*.

No obstante, aunque utilizando PyCharm, la forma anterior es la más simple, también se pueden realizar estas funciones desde la terminal del sistema [19]:

1. Entrar en el directorio contenedor del proyecto.

2. Creación de un subdirectorio *.git* con todos los archivos necesarios para la conexión entre repositorio y plataforma.

```
# git init
```

3. Indicación y confirmación de los archivos a subir al repositorio.

```
# git add [file]
# git commit
```

4. Actualización de la versión de los archivos del ordenador local al repositorio.

```
# git push origin master
```

5. Actualización de la versión de los archivos del repositorio al ordenador local.

```
# git pull origin master
```

Por último, es fundamental destacar que GitHub permite publicar en un sitio web el contenido del repositorio de forma completamente gratuita, utilizándose esta opción para la creación del dominio web donde se aloja el tutorial [6]:

`https://victorffdez.github.io/Python_Biochemistry_Tutorial/`

4.1.3. Markdown

Markdown es un lenguaje de marcaje ligero creado por John Gruber y Aaron Swartz en 2004, que permite la escritura rápida de texto. Fue diseñado con el objetivo de ser tan fácil de leer y escribir como fuera posible, presentando la gran ventaja de poder convertir el documento de texto a formato HTML [14].

Todas las secciones del tutorial han sido desarrolladas utilizando el lenguaje Markdown, siendo cada uno de los apartados un documento de texto en formato *.md*, característico de Markdown. En el Apéndice C se muestra de ejemplo uno de estos ficheros Markdown creados. En este fichero se puede comprobar que este lenguaje presenta una sintaxis particular en la que el asterisco (*) y el guión bajo (_) se utilizan para indicar cursiva y/o negrita, la almohadilla (#) para determinar los encabezados y el acento grave (‘) para identificar código.

4.1.4. Mkdocs

Mkdocs es un software que permite generar sitios estáticos, orientado a la creación de plataformas de documentación [4]. Esta herramienta escrita en Python es bastante simple de configurar, presentando como principal requisito que los ficheros estén en formato *.md* de Markdown.

Mkdocs ha sido utilizado en el desarrollo de este tutorial con el objetivo de poder enriquecer el documento utilizando las extensiones que posibilita, como incluir notas, mejorar la visualización del bloque de código o insertar diagramas. Sumado a esto, al generar sitios estáticos, esta herramienta permite visualizar en todo momento las modificaciones realizadas en el texto.

Al estar en código Python, la instalación de Mkdocs es muy simple y se puede realizar desde la opción *Python Packages* de PyCharm. Desde aquí, ha de instalarse el paquete *Mkdocs*, así como el paquete del tema que se desea emplear, en este caso *Mkdocs-material* [9]. Por último y de forma complementaria, también se instala el paquete *Mkdocs-material-extension* que contiene algunas funciones adicionales.

- Visualización del sitio web desde el directorio de trabajo.

```
# mkdocs serve
```

- Construcción de la documentación del sitio web al finalizar el proyecto.

```
# mkdocs build
```

Por tanto, el aspecto visual del sitio web del tutorial depende principalmente de la sintaxis de los ficheros *.md*, así como de la configuración del tema de Mkdocs seleccionado, albergada en un fichero *mkdocs.yml* mostrado en el Apéndice B.

4.1.5. Google Colab

La forma tradicional de escribir y ejecutar un programa en Python es crear un archivo *.py* que almacene su código fuente. No obstante, aunque en la mayor parte del tutorial sea este el método utilizado, en el apartado *Ejercicios* de la sección *Graficación* se introduce al alumno en el empleo de los cuadernos de Google Colab [2].

Los cuadernos en lenguaje de programación (normalmente conocidos como *notebook*) permiten combinar texto y código, siendo muy útiles para documentar el trabajo a la vez que se prueba código fácilmente. En este caso, Google Colaboratory es un producto de Google Research que permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador [2]. En el Apéndice A.2 se muestra parte del código de este cuaderno.

4.1.6. LaTeX

LaTeX [1] es un sistema de composición de textos de alta calidad, cuyo uso está altamente implementado en la producción de documentación técnica y científica. Este sistema utiliza comandos TeX de código abierto, especialmente útil para la escritura científica de artículos de revistas, informes técnicos, libros y presentaciones de diapositivas.

Esta memoria ha sido desarrollada completamente en LaTeX, aunque en lugar de utilizar el software, se ha utilizado la herramienta Overleaf [12], un editor online de LaTeX fácil de usar, que no necesita instalación ni actualizaciones, y permite trabajar en el proyecto desde cualquier dispositivo. Además, Overleaf incluye la opción *Auto Compile*, produciendo una salida en interfaz totalmente compilada a medida que se escribe, por lo que la visualización del resultado es inmediata.

4.2. Desarrollo: organización y visualización

El tutorial está dividido en un total de doce secciones para un aprendizaje óptimo, partiendo desde conceptos básicos de programación en Python hasta adentrar al alumnado en conceptos de una mayor complejidad. Estas secciones no serán independientes unas de otras, siguiendo una metodología de trabajo continua en la que el alumno debe conocer los conceptos de secciones anteriores para poder seguir avanzando a lo largo del tutorial.

En cada uno de estos apartados, sumado a las explicaciones teóricas simples, se incluyen múltiples ejemplos y ejercicios prácticos con soluciones ocultas, que convertirán el tutorial en una herramienta muy interactiva. En el apartado 4.3 de este trabajo se muestran algunos de los ejemplos y ejercicios propuestos, e información acerca de los enlaces de interés que aparecen al final de cada sección.

En relación con la parte visual del tutorial, en la figura 4.1 se muestra la visualización de la página principal, el apartado *Inicio*.

Figura 4.1: Aspecto visual de la página principal del tutorial.

En esta figura se observa:

- A la izquierda aparece el índice del tutorial, con todas las secciones que alberga. En color se representa aquella sección en la que se encuentra el usuario.
- A la derecha aparece la tabla de contenido de la sección actual. Al igual que en el índice general, aparecerá coloreada la subsección en la que esté el usuario, actualizándose a medida que avanza.
- En la parte superior aparece el título del tutorial, que se verá sustituido por el nombre del apartado en el que se encuentre el usuario; así como una barra de búsqueda y un enlace al repositorio GitHub del sitio web.

Como se puede ver en la figura 4.1, el color establecido del sitio web, así como de los hiperenlaces insertados, es el azul. No obstante, el usuario puede cambiar a modo oscuro en el interruptor situado al lado del buscador. En la figura 4.2 se puede observar este modo oscuro, así como algunas palabras o expresiones destacadas en el texto en formato negrita, cursiva y subrayado. Sumado a esto, en la parte superior de esta figura se observa un botón de *Volver al principio*, que le permite realizar al usuario esta acción de forma cómoda.

A lo largo del tutorial también se incluirán una serie de tablas e imágenes, destacando su uso principalmente en el apartado *Graficación* para poder mostrar las gráficas obtenidas a través de la línea de comandos. En la figura 4.3 se observa una gráfica correspondiente al subapartado *Formato a la figura*, apareciendo centrada y estructurada de forma aclarativa con el texto.

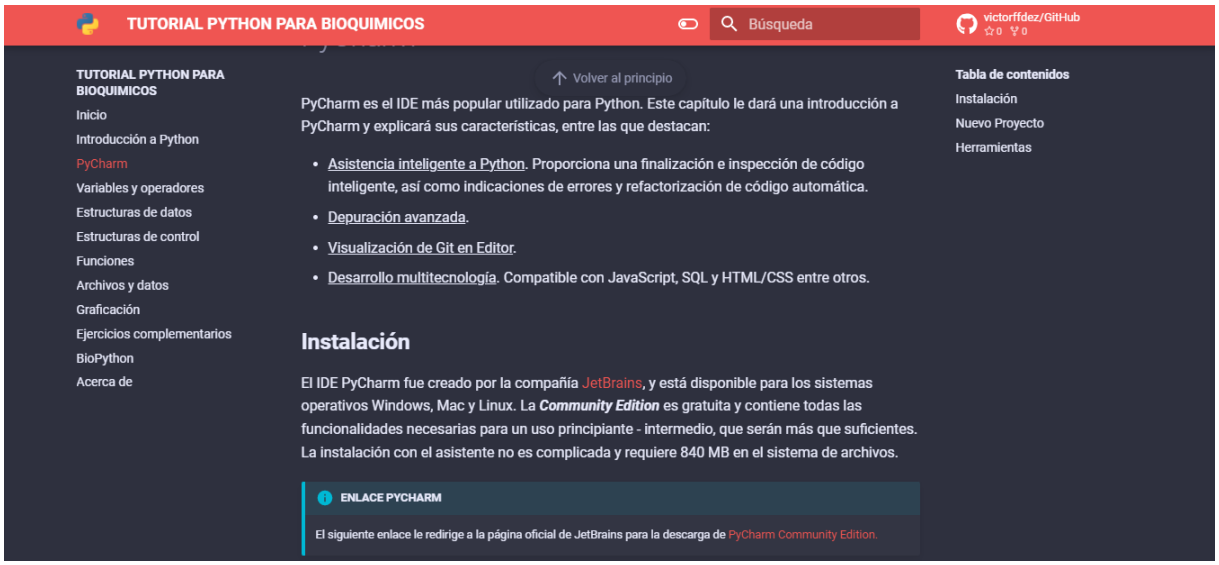

Figura 4.2: Aspecto visual en modo oscuro del apartado *PyCharm*.

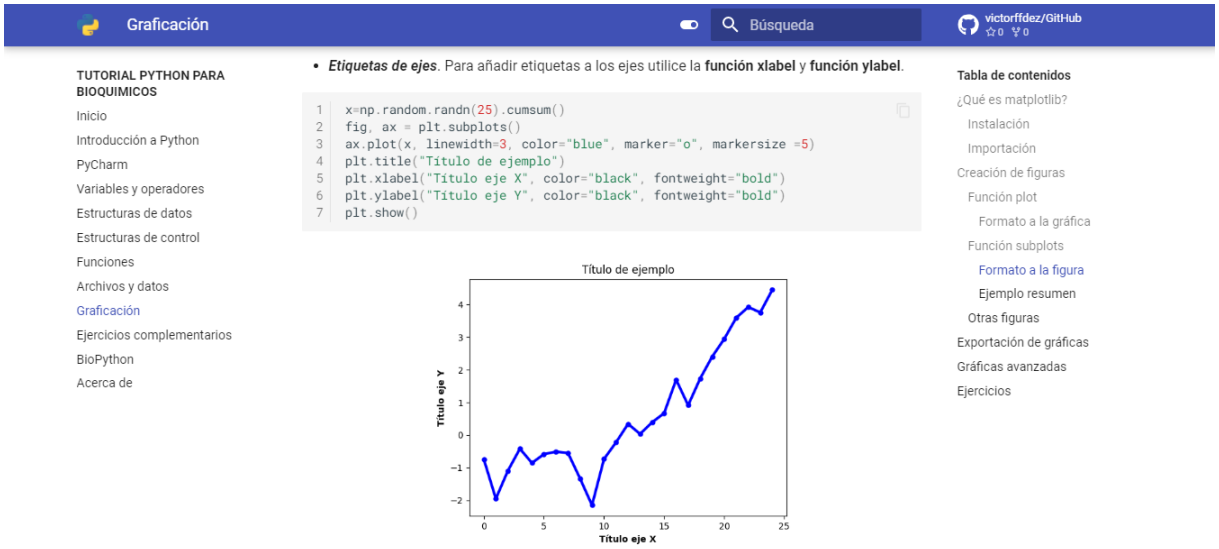

Figura 4.3: Aspecto visual de las imágenes insertadas.

En la figura 4.3, tal como se puede observar la visualización de las imágenes en la página web, se ve en la parte superior de la imagen el formato que presentan las líneas de código de programación. Markdown permite incluir todas las líneas de código en un formato diferente al texto normal, facilitando su visualización. Sumado a esto, este código incluye la opción de copia al portapapeles en el borde superior derecho, siendo una función muy útil para que el alumno pueda copiar ejemplos íntegros y los practique en su IDE personal. El formato especial que presenta el código se puede observar de forma más clara en la figura 4.4, correspondiente al apartado *Biopython*, esta vez en modo oscuro.

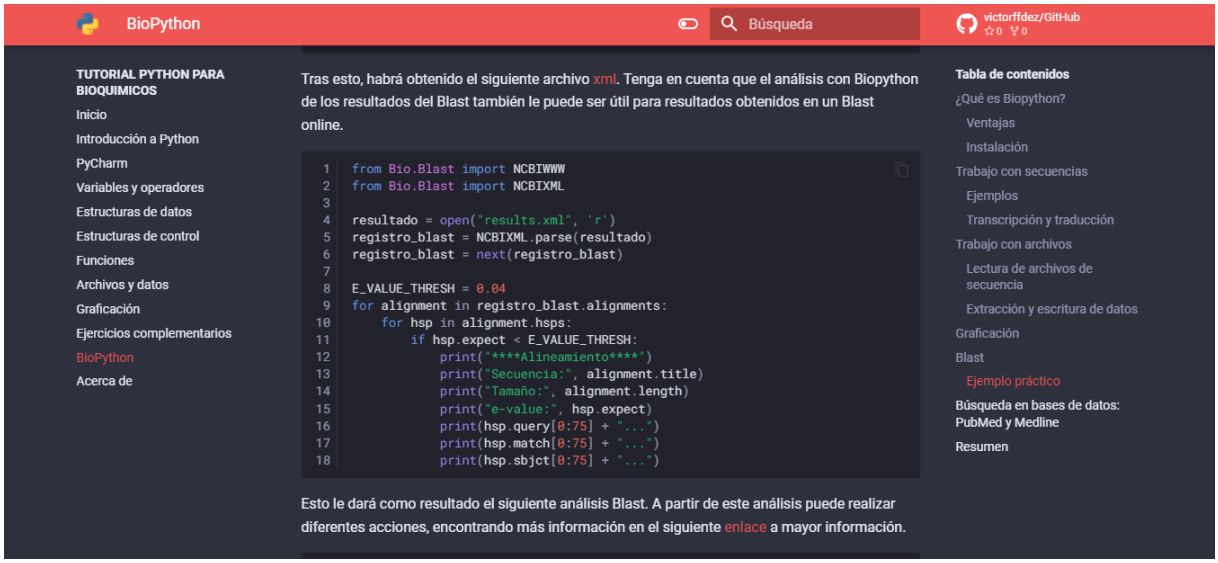

Figura 4.4: Aspecto visual del código de programación.

Markdown, al igual que permite incluir bloques de código de forma diferenciada, también permite incluir otros bloques de texto, muy útil para incluir bloques como *Notas*, *Ejemplos*, *Ejercicios*, *Información adicional* y *Enlaces de interés*. El uso de estos bloques facilita la división y visualización del contenido de la página web, y por ello aparecen en todas las secciones del tutorial. En la figura 4.5 aparecen representados cuatro bloques diferentes:

1. Respuesta. Bloque de texto que oculta su contenido en forma de desplegable.
2. Bloque de texto normal.
3. Diccionarios. Bloque de texto en formato nota, que alberga bloque de código, ya que se pueden combinar varios bloques diferentes como se desee.
4. Enlaces de interés.

Como se observa en la figura 4.6, al final de cada sección aparece una serie de enlaces de interés para que el alumnado puedo ampliar el contenido visto. Además,

Figura 4.5: Aspecto visual de los bloques de texto.

durante el texto también se incluyen hipervínculos que le serán de ayuda. A la derecha de dicha figura aparece una flecha que permite avanzar y retroceder a lo largo de las secciones, función que también se puede realizar en el índice de contenido. Por último, en la zona inferior de la figura 4.6 aparece información relacionada con el *Copyright* del sitio web y enlaces vinculados con las redes sociales de la Universidad de Córdoba y Python.

Figura 4.6: Aspecto visual del final de la sección *Inicio*.

Todas las figuras que aparecen hasta el momento han estado relacionadas con la visualización del sitio web desde el ordenador. No obstante, el diseño web es totalmente compatible con dispositivos móviles, siendo por tanto un diseño web adaptativo (conocido en su término inglés como *responsive web design* [11].) Los diseños web

adaptativos engloban a aquellos sitios web capaces de adaptarse y optimizarse al dispositivo utilizado para su visualización, ya sean teléfonos móviles inteligentes, tabletas, etc.

En la figura 4.7 se muestra la visualización de la página principal desde un teléfono móvil inteligente o *smartphone*.

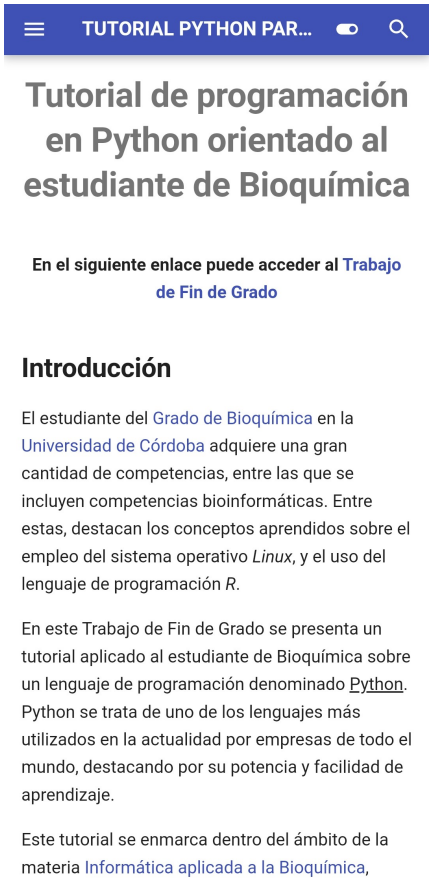

Figura 4.7: Aspecto visual de la página principal desde un teléfono inteligente.

Todo el contenido de la página web se adapta de forma óptima a su visualización. Como se observa en la figura 4.7, la barra de la parte superior se ve reducida a tres líneas que mostrarán en un desplegable el índice de secciones del tutorial, el interruptor para cambiar a modo oscuro y la lupa de búsqueda. Comparado con la figura 4.1, se observa cómo en la barra del dispositivo móvil ha desaparecido el enlace al repositorio de GitHub, que pasa a encontrarse en el desplegable del índice de contenido.

De forma adicional, en la figura 4.8 también se puede observar la adaptación del contenido del sitio web. En esta figura, en la imagen A se observa una tabla correspondiente a la sección *Introducción a Python* en modo oscuro; y en la imagen B un gráfico correspondiente a la sección *Graficación*. En esta última figura se puede apreciar cómo se adapta el bloque de código a la visualización desde el móvil, ya que en lugar de minimizar la letra, incluye una barra de desplazamiento que permite ver el código de forma mucho más clara y precisa.

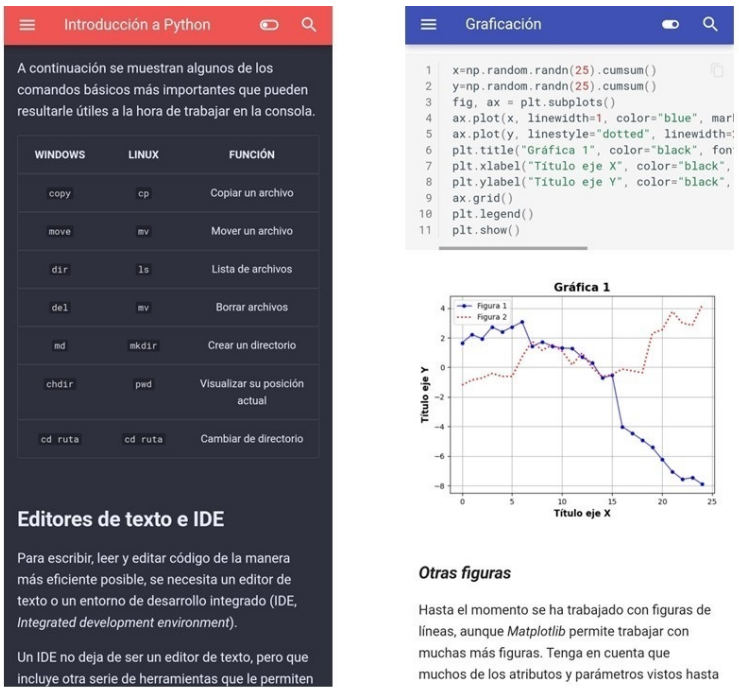

Figura 4.8: Comparativa entre el aspecto visual en modo oscuro (A) y claro (B) desde un teléfono móvil inteligente.

A continuación se describe de forma general cada una de las doce secciones que componen el tutorial.

4.2.1. Inicio

La página *Inicio* es la página principal del tutorial, y por tanto, la primera que se muestra por pantalla al abrirlo en el navegador. En esta página, el alumnado en primer lugar puede acceder a este documento del Trabajo de Fin de Grado, por si desea leerlo y conocer más sobre la realización del sitio web del tutorial. Está compuesta principalmente por una introducción acerca del tutorial de programación en Python, en la que se explica el contexto docente en el que está situado el trabajo y se destaca la correlación entre la Bioquímica y el mundo de la Bioinformática.

En la elaboración de este apartado ha sido de gran ayuda el artículo "All biology is computational biology" [16].

4.2.2. Introducción a *Python*

Esta es la primera sección del tutorial enfocada de forma particular en Python. En primer lugar, se explica al alumno en qué consiste Python, es decir, un lenguaje de programación creado por Guido van Rossum en 1990 y actualmente desarrollado y mantenido por la *Python Software Foundation* [23]. Además, se muestra una serie de

características de este lenguaje que lo convierten en una gran opción para trabajar en el mundo de la programación. Posteriormente, esta sección muestra cómo realizar la instalación de Python en Linux, MacOs y Windows. En la mayoría de sistemas Linux y MacOs Python viene instalado de fábrica, por lo que se explica cómo comprobar de qué versión se trata, por si estuviera desfasada y así poder actualizarla. En Windows, al no estar instalada en el sistema, se muestra cómo instalar e iniciar Python desde la terminal [17]. Una vez instalado, se va a crear y ejecutar el programa más simple en el campo de la Informática, un programa que salude al mundo. De forma complementaria, también se muestran algunos de los comandos básicos que le pueden resultar útiles al alumno a la hora de trabajar desde la consola de Linux y Windows. Por último, se describen qué son los editores de texto e IDE, para introducir de forma breve al alumno en el siguiente apartado.

En el desarrollo de esta sección ha sido de especial relevancia el libro ”Curso intensivo de Python, 2a edición: introducción práctica a la programación basada en proyectos” [17] y "Python 3 Reference Manual" [23].

4.2.3. PyCharm

Una vez explicado en la sección anterior qué son los editores de texto e IDE, este apartado está íntegramente especializado en PyCharm. Este es el IDE recomendado al alumno para el seguimiento del tutorial, y por este motivo se explica en primer lugar cómo instalarlo en su ordenador. Una vez instalado, se muestra cómo crear un nuevo proyecto desde la primera ejecución. Tras crear el proyecto, se describen de forma detallada las principales funcionalidades de los elementos que componen la interfaz de PyCharm, mostrando numerosas imágenes de la descomposición por partes de dicha interfaz. Es importante destacar que en esta sección el alumno encontrará información acerca de la instalación y manejo de paquetes en PyCharm, así como de Git y GitHub.

Para la elaboración de esta sección ha sido de gran utilidad la información proporcionada en la página oficial de Overleaf [12] y el artículo "Introduction to Git & GitHub" [6].

4.2.4. Variables y operadores

En este apartado se comienza con la introducción del alumno en el lenguaje Python, en este caso en el empleo de variables y operadores, sin duda pilares básicos en el conocimiento de cualquier lenguaje de programación. Para ello se define qué son las variables, cómo se nombran y utilizan, y qué tipos diferentes de variables se pueden encontrar. Posterior a la definición de variables en Python, se explica qué son los operadores, describiéndose los cuatro tipos principales de operadores: aritméticos, relacionales, de asignación y lógicos. Para ilustrarlo de forma más clara, se utilizan tablas que relacionan el tipo de operador, su nombre y su función.

Para poder desarrollar tanto este apartado como los siguientes, han sido de gran ayuda los libros "Python 3 Reference Manual" [23] y "Curso intensivo de Python, 2a edición: introducción práctica a la programación basada en proyectos" [17], y el curso "Aprende python" [20].

4.2.5. Estructuras de datos

El alumno, una vez conozca y se maneje en el uso de variables y operadores, irá en este apartado un paso más allá, introduciéndose en el empleo de estructuras de datos. Estas estructuras son una colección de datos caracterizadas por su organización y las operaciones definidas en ellos, divididas en tres tipos: listas, tuplas y conjuntos [23]. Por tanto, en esta sección se muestra todo lo necesario para trabajar con estas estructuras de datos, poniendo especial énfasis en el uso de las listas, ya que son una de las estructuras más utilizadas en lenguaje de programación. Por este motivo, el alumno aprenderá a crear una lista; acceder, cambiar, añadir y eliminar elementos; así como algunas funciones específicas en el trabajo con dichas listas. Por último, se introduce al alumno en el uso de diccionarios, un último tipo de estructura de datos avanzado cuyo uso le puede resultar de interés.

4.2.6. Estructuras de control

Las estructuras de control permiten modificar el flujo de ejecución de instrucciones de un programa, dependiendo del resultado de unas condiciones [17]. En esta sección, se van a describir las tres estructuras de control características de cualquier lenguaje de programación: bucles FOR, condicional IF y bucles WHILE. En primer lugar se describirán los bucles FOR, introduciendo al alumno a la sintaxis y trabajo con estos bucles. En segundo lugar se explicará el uso del condicional IF, así como de las sentecias IF, IF-ELSE, IF-ELIF-ELSE. Por último se describirán los bucles WHILE, mostrando las diferencias respecto a los bucles FOR en su sintaxis y uso.

4.2.7. Funciones

En este apartado el alumno aprenderá a trabajar con funciones en Python, definiendo como función un bloque de código diseñado para realizar una tarea utilizable en el momento que se desee. Desde la creación de la función más simple, una que permita saludar al usuario por pantalla; en esta sección se muestra cómo llamar a una función, cómo modificar sus parámetros y argumentos y qué funciones definidas existen ya en Python. Por último también se describe el uso de módulos, archivos en los que se pueden almacenar las funciones para importarse posteriormente.

4.2.8. Archivos y datos

En la mayoría de ocasiones el alumno necesitará trabajar con archivos y datos en Python. Por ese motivo, en este apartado se describen una serie de conceptos básicos relacionados con la lectura y manipulación de archivos, y la importación/exportación de datos en formato csv, json, excel y sql, utilizando las librerias NumPy y Pandas.

4.2.9. Graficación

Esta sección del tutorial está aplicada a la realización de gráficas mediante Python. Para esta tarea, en Python existen una gran cantidad de librerías disponibles, como Numpy, ya vista en la sección anterior para el tratamiento de datos. No obstante, la librería más utilizada por todo el ámbito científico es matplotlib. El paquete matplotlib es relativamente grande y complejo, y entre otros contiene dos módulos principales, pylab y pyplot, siendo este último el que se describe en el tutorial. El alumno en este apartado aprenderá a crear diversas figuras con la función plot y subplot, como gráficas de líneas, de dispersión, diagramas de barras, de sectores e histogramas. Una vez creadas, se muestran numerosas opciones para modificar estas figuras, personalizando el formato, y exportarlas. Por último se introduce al alumno en el uso de gráficas avanzadas, figuras de un nivel de manejo de Python superior.

Este apartado ha sido desarrollado utilizando principalmente la información proporcionada en la guía de usuario de la página oficial de Matplotlib [22].

4.2.10. Ejercicios complementarios

Una vez llegado a este punto, el alumno debe haber adquirido los suficientes conocimientos básicos que le permitan trabajar en Python. Por ese motivo, en este apartado se proponen un total de ocho ejercicios reales del campo de la Bioquímica y Biología Molecular, relacionados con la manipulación de fragmentos nucleotídicos y/o proteicos, para que practique y compruebe sus conocimientos en este lenguaje de programación.

4.2.11. Biopython

Esta sección esta dedicada al uso de Biopython, el paquete de Python más utilizado en el ámbito de la Biología Computacional, que contiene una gran cantidad de herramientas útiles en Bioinformática. Esta librería fue creada en el año 1999 por Brad Chapman y Jeff Chang, y actualmente está soportado por el *Proyecto Biopython* [10]. En esta sección se muestra al alumno cinco modos de trabajo diferentes con esta librería:

- Trabajo con secuencias. Se incluyen ejemplos de conteo de nucleótidos, cálculo de porcentajes G+C, transcripción y traducción a partir de secuencias nucleotídicas.

- Trabajo con archivos. Se propone un archivo FASTA que contiene todas las entradas resultantes de una búsqueda en la base de datos nucleotide de NCBI. A partir de este archivo, se muestra como poder leer, extraer y modificar datos.

- Graficación. A partir del archivo FASTA propuesto en la sección anterior, se realizan representaciones gráficas del porcentaje G+C y del tamaño de las secuencias nucleotídicas.

- Blast. Se propone un nuevo archivo FASTA, que en este caso contiene la secuencia codificante de un gen en dos especies diferentes. A partir de estas secuencias, se describe cómo realizar un Blast desde Python.

- Búsqueda en bases de datos: PubMed y Medline. En esta sección se muestra cómo realizar búsquedas y obtener información de las principales fuentes de datos en el campo de la Bioquímica, sin necesidad del navegador web.

  Para la elaboración de este apartado ha sido de gran utilidad la información y documentación proporcionada en la página oficial de Biopython [10].

Figura 4.9: Logo del Proyecto Biopython.

4.2.12. Acerca de

En este apartado simplemente se indica que el tutorial de programación en Python ha sido desarrollado por Víctor J. Fernández Ramírez y dirigido por el profesor Manuel J. Marín Jiménez como Trabajo de Fin de Grado de la titulación de Grado de Bioquímica en la Universidad de Córdoba.

4.3. Material de apoyo

Durante el desarrollo del tutorial, de forma complementaria a las diferentes explicaciones teóricas que componen cada apartado, se le proporciona al alumnado material de apoyo que facilite el entendimiento de dichas explicaciones, en forma de ejemplos, ejercicios y enlaces de interés.

4.3.1. Ejemplos prácticos

Todas las secciones que componen el tutorial contienen multitud de ejemplos prácticos, que convierten el tutorial en una herramienta práctica y dinámica, evitando la acumulación de conceptos teóricos difíciles de asimilar por el alumno. Por ese motivo, toda la teoría desarrollada viene acompañada de ejemplos que faciliten la comprensión. A continuación se muestran algunos de los ejemplos que se pueden encontrar en el tutorial:

```
#TRANSCRIPCION MEDIANTE BIOPYTHON
from Bio.Seq import Seq
seq_codificante = Seq("AATGCGGCTTAACCACAC")
seq_molde = seq_codificante.reverse_complement()
ARN_mensajero = seq_codificante.transcribe()
print(ARN_mensajero)

#TRADUCCION MEDIANTE BIOPYTHON
from Bio.Seq import Seq
seq_codificante = Seq("AATGCGGCTTTACCACAC")
seq_molde = seq_codificante.reverse_complement()
ARN_mensajero = seq_codificante.transcribe()
proteina= ARN_mensajero.translate()
print(proteina)
```

4.3.2. Ejercicios

Para que el alumno pueda practicar y poner a prueba los conocimientos que acaba de adquirir, al final de cada sección del tutorial se proponen una serie de ejercicios. Además, para que exista una mayor disponibilidad de ejercicios, una sección del tutorial está íntegramente compuesta por ejercicios, en los que el alumno pondrá en práctica la mayoría de conceptos aprendidos en las secciones anteriores. En esta sección todos los ejercicios están enfocados en el campo de la Bioquímica y la Biología Molecular.

Gracias a la sintaxis de Markdown, la respuesta a estos ejercicios está oculta al alumnado, para que previamente pueda realizarlos desde su propio entorno PyCharm

y posteriormente compruebe la respuesta. A continuación se muestra uno de los ejercicios propuestos en el tutorial:

```
ENUNCIADO: Defina una funcion que muestre la posicion de el primer
   nucleotido que constituye el codon de inicio en un fragmento de
   ADN.

RESPUESTA:
def codon_inicio(segmento_DNA):
    for i in range(len(segmento_DNA)):
        triplete = segmento_DNA[i:i+3]
        if triplete == 'ATG':
            return(i+1) # Ya que la posicion 1 es 0
    else:
        return ("No hay codon de inicio.")
```

4.3.3. Enlaces de interés

Al final de cada sección aparecen una serie de enlaces de interés para que el alumnado puedo ampliar el contenido visto. Entre estos enlaces, el alumno va a encontrar el propio material utilizado para la realización del sitio web, así como material complementario que le pueda resultar de interés o que le ayude a ampliar sus conocimientos acerca de la materia. Sumado a esto, durante el texto también se incluyen hipervínculos que le serán de ayuda.

Capítulo 5

Conclusiones y trabajo futuro

Como se puede apreciar en este Trabajo de Fin de Grado, el desarrollo del mismo ha requerido el empleo de una gran diversidad de herramientas, algunas utilizadas, aunque la mayoría no, durante el Grado de Bioquímica. El aprendizaje adquirido en la asignatura *Informática aplicada a la Bioquímica* se ha puesto en práctica durante la realización, ya que en esta asignatura se adquieren conocimientos básicos sobre el manejo de lenguajes de programación y el sistema de edición de textos LaTeX. Esta asignatura impartida en el Grado de Bioquímica de la Universidad de Córdoba introduce al alumno en la programación en *R*, obteniendo una serie de conocimientos de programación muy útiles a la hora de aprender Python. Sumado a esto, los conocimientos adquiridos sobre manejo de la terminal también han sido de vital importancia.

No obstante, durante el Grado de Bioquímica la asignatura *Informática aplicada a la Bioquímica* no ha sido la única en la que se ha utilizado programación. En la asignatura *Biología Molecular de Sistemas*, en la que los alumnos adquieren conocimientos en el campo de la Bioinformática, también se realiza programación en R. En base a esto, se detectó una necesidad de aprendizaje de lenguajes alternativos a R, destacando un lenguaje muy popular como Python, que empieza a competir directamente con R en el análisis de datos en el área de la Biología Computacional. Esta ha sido la principal razón que ha fomentado la realización de este proyecto, permitiendo que el alumno del Grado de Bioquímica, u otros estudios del ámbito científico, amplíe sus conocimientos en programación.

Para el estudiante de Bioquímica también supone un gran beneficio que la realización de este tutorial haya sido por un estudiante propio de su Grado, ya que conoce ampliamente las dificultades en el campo de la Informática a las que se puede enfrentar el alumno. Esto le ha permitido al autor crear un tutorial con mayor accesibilidad por el total del alumnado, enfocado en los conocimientos convenientes. Debido a esto, el autor del Trabajo de Fin de Grado se ha tenido que poner en la piel del profesorado, buscando la creación de un tutorial con el nivel de complejidad suficiente para que el alumno obtenga los conocimientos adecuados sin llegar a perder la motivación. Por este motivo, se incluyen numerosos ejemplos del campo de la Bioquímica y Biología

Molecular, aspecto que convierte el tutorial en una herramienta muy útil, restándole complejidad para el alumnado y convirtiéndolo en un tutorial diferencial.

Como se ha destacado anteriormente, Python no ha sido la única herramienta, aunque sí la principal, con la que el autor de este trabajo ha tenido que familiarizarse. Durante el desarrollo se han empleado tecnologías como GitHub, Mkdocs, Markdown y Google Colab, las cuales no han sido vistas durante el Grado y su manejo ha sido necesario para la realización del trabajo. El conocimiento adquirido sobre estas herramientas es de gran relevancia, ya que algunas como GitHub y Google Colab tienen un notable uso en el ámbito científico.

En cuanto al aprendizaje en Python, el autor ha tenido que aprender de cero y consolidar una serie de conocimientos básicos en este lenguaje de progamación que le permitan transmitírselos de una forma certera al alumnado. Además, el tutorial incluye una sección sobre *Biopython*, herramienta descrita en el presente trabajo en el apartado 4.2.11, que ofrece al alumno un nivel más avanzado en el uso de Python. De hecho, esta sección le introduce en funciones reales que puede realizar un bioquímico mediante Python, como alineamiento de secuencias, búsquedas en bases de datos y análisis de archivos FASTA de secuencias genómicas.

En base a esto, los objetivos propuestos al inicio de este TFG se han cumplido: se ha desarrollado un tutorial de Python a nivel básico, para que cualquier estudiante de Bioquímica pueda manejarse en el mismo; a la vez que se han introducido una serie de conceptos más avanzados que permitan al alumno enfrentarse a problemas reales del mundo de la Bioinformática.

Por último, tras la realización de este proyecto, como trabajo futuro se deja la puerta abierta a la realización de un tutorial de nivel más avanzado que pueda adentrarse de una forma más detallada en las funciones descritas y en otras no mencionadas. Sumado a esto, también se ha incluido trabajo en forma de ejercicios resueltos, tanto en scripts de Python como en cuadernos de Google Colab. No obstante, este tutorial ha sido incluido dentro del proyecto UCOdemy dirigido al aprendizaje de lenguajes de programación, por lo que siempre va a poder ser actualizado y expandido si así es necesario. De esta manera, el tutorial se puede renovar a medida que evoluciona el lenguaje de programación Python y su uso en el campo de la Biología Molecular.

Conclusions and future work

As can be seen in this Final Degree Project, its development has required the use of a wide variety of tools, some used, although most not, during the Biochemistry Degree. The learning acquired in the subject *Computer science applied to Biochemistry* has been put into practice during the course, since in this subject basic knowledge is acquired about the use of programming languages and the LaTeXtext editing system. This subject taught in the Degree in Biochemistry at the University of Córdoba introduces the student to programming in *R*, obtaining a series of very useful programming knowledge when learning Python. In addition to this, the knowledge acquired on handling the terminal has also been of vital importance.

However, during the Biochemistry Degree, the subject *Computer Science applied to Biochemistry* was not the only one in which programming was used. In the subject *Molecular Systems Biology*, in which students acquire knowledge in the field of Bioinformatics, programming in R is also carried out. Based on this, a need to learn alternative languages to R was detected, highlighting a very popular language such as Python, which is beginning to compete directly with R in data analysis in the area of Computational Biology. This has been the main reason that has promoted the realization of this project, allowing the student of the Degree in Biochemistry, or other studies in the scientific field, to expand their knowledge in programming.

For the Biochemistry student, it is also a great benefit that this tutorial has been carried out by a student of his own Degree, since he is fully aware of the difficulties in the field of Computer Science that the student may face. This has allowed the author to create a tutorial with greater accessibility for all students, focused on the appropriate knowledge. Due to this, the author of the Final Degree Project has had to put himself in the shoes of the teaching staff, seeking to create a tutorial with a sufficient level of complexity so that the student obtains the appropriate knowledge without losing motivation. For this reason, numerous examples from the field of Biochemistry and Molecular Biology are included, an aspect that makes the tutorial a very useful tool, reducing complexity for students and turning it into a differential tutorial.

As previously highlighted, Python has not been the only tool, although it has been the main one, with which the author of this work has had to become familiar. During the development, technologies such as GitHub, Mkdocs, Markdown and Google Colab have been used, which have not been seen during the Degree and their management

has been necessary to carry out the work. The knowledge acquired about these tools is highly relevant, since some such as GitHub and Google Colab are widely used in the scientific field.

Regarding learning in Python, the author has had to learn from scratch and consolidate a series of basic knowledge in this programming language that allow him to transmit it accurately to the students. In addition, the tutorial includes a section on *Biopython*, a tool described in this work in the section 4.2.11, which offers the student a more advanced level in the use of Python. In fact, this section introduces you to actual functions that a biochemist can perform using Python, such as sequence alignment, database searches, and analysis of FASTA files of genomic sequences.

Based on this, the objectives proposed at the beginning of this TFG have been fulfilled: a Python tutorial has been developed at a basic level, so that any Biochemistry student can handle it; At the same time, a series of more advanced concepts have been introduced that allow the student to face real problems in the world of Bioinformatics.

Finally, after the completion of this project, as future work, the door is left open to the realization of a more advanced level tutorial that can go into a more detailed way in the functions described and in others not mentioned. In addition to this, work has also been included in the form of solved exercises, both in Python scripts and in Google Colab notebooks. However, this tutorial has been included in the UCOdemy project aimed at learning programming languages, so it can always be updated and expanded if necessary. In this way, the tutorial can be renewed as the Python programming language and its use in the field of Molecular Biology evolve.

Bibliografía

[1] *Latex: A Document Preparation System, 2/E*. Pearson Education, 1994.

[2] Ekaba Bisong. *Google Colaboratory*, pages 59–64. 09 2019.

[3] Díaz-Ricardo Y. Challenger-Pérez, I. and R.A. Becerra-García. El lenguaje de programación python. ciencias holguín, 20 (2), 1-13, 2014.

[4] Tom Christie. Mkdocs. project documentation with markdown, 2014.

[5] Nello Cristianini and Matthew W Hahn. *Introduction to computational genomics: a case studies approach*. Cambridge University Press, 2006.

[6] J. Cristóbal. *Crea páginas estáticas con GitHub: De manera ilimitada y gratuita*. Javier Cristóbal, 2015.

[7] Universidad de Córdoba. Grado en bioquímica. `https://www.uco.es/ciencias/es/grado-bioquimica`. Acceso: 2022/05/10.

[8] Universidad de Córdoba. Grado en bioquímica, competencias básicas. `http://www.uco.es/organiza/centros/ciencias/es/competencias-bioquimica`. Acceso: 2022/05/11.

[9] Martin Donath. Material for mkdocs. `https://squidfunk.github.io/mkdocs-material/`. Acceso: 2022/05/02.

[10] Brad Chapman et al. Biopython tutorial and cookbook. `http://biopython.org/DIST/docs/tutorial/Tutorial.html`. Acceso: 2022/06/01.

[11] Brett S Gardner. Responsive web design: Enriching the user experience. *Sigma Journal: Inside the Digital Ecosystem*, 11(1):13–19, 2011.

[12] John Hammersley and John Lees-Miller. Overleaf, editor de latex online. `https://www.overleaf.com`. Acceso: 2022/05/24.

[13] Manuel Jesús Marín Jiménez. *Guía Docente Informática aplicada a la Bioquímica*. Universidad de Córdoba, Curso 2020-2021.

[14] The Daring Fireball Company LLC. Markdown: Syntax. `https://daringfireball.net/projects/markdown/syntax`. Acceso: 2022/05/24.

[15] Nicholas M Luscombe, Dov Greenbaum, and Mark Gerstein. What is bioinformatics? an introduction and overview. *Yearbook of medical informatics*, 10(01):83–100, 2001.

[16] Florian Markowetz. All biology is computational biology. *PLoS biology*, 15(3):e2002050, 2017.

[17] E. Matthes and B.P. González. *Curso intensivo de Python, 2a edición: introducción práctica a la programación basada en proyectos*. Anaya Multimedia, 2020.

[18] CA Michael Kennedy. Pycharm, python ide for professional developers, 2017.

[19] Pablo Orviz Fernández. Introduction to git & github. 2022.

[20] Sergio Delgado Quintero. *Aprende Python*. GNU General Public License v3.0, 01/04/2022.

[21] M. Schultz and T. Ebenhard. La agenda 2030 y los ecosistemas - un documento para la discusión acerca de los vínculos existentes entre las metas de aichi para la diversidad biológica y los objetivos de desarrollo sostenible. *SwedBio, Stockholm Resilience Centre*, 2016.

[22] John Hunter & the matplotlib development team. Matplotlib, release 3.5.1. `https://matplotlib.org/`. Acceso: 2022/06/01.

[23] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[24] Jonathan Yates. Python: Practical python programming for beginners and experts (beginner guide), 2016.

Apéndice A

Material digital de apoyo

A.1. Python script

Como se ha mencionado en el apartado 4.3.2, a lo largo del tutorial se proponen una serie de ejercicios prácticos para que el alumno compruebe y desafíe su aprendizaje. A continuación se muestra una parte del script de Python perteneciente a la sección 4.2.10; script denominado *script_ejercicios.py*, que el alumno puede descargarse en esta sección del tutorial y que almacena en un script todos los ejercicios propuestos.

```
1  #Imprima por pantalla todos los codones que comiencen por timina-T.
2
3  bases = ['A', 'T', 'C', 'G']
4  print('Codones que empiezan por T:')
5  for segunda_base in bases:
6      print('Codones que empiezan por T' + segunda_base)
7      for tercera_base in bases:
8          print('T' + segunda_base + tercera_base)
9
10 #Determine el porcentaje de aminoacidos de un fragmento proteico
       almacenado como cadena.
11 #Se propone de ejemplo el siguiente fragmento:
       PCCWWLAKVRMIKGEFYVIEYAACD
12
13 secuencia_proteica ="PCCWWLAKVRMIKGEFYVIEYAACD"
14 aminoacidos = set(secuencia_proteica)
15 secuencia_length = len(secuencia_proteica)
16 for aminoacido in aminoacidos:
17     aminoacido_count = secuencia_proteica.count(aminoacido)
18     aminoacido_porcentaje = (aminoacido_count/secuencia_length) *
           100
19     print(aminoacido,":",round(aminoacido_porcentaje,1))
```

A.2. Google Colab

En la mayor parte del tutorial se utilizan los script de Python para escribir y ejecutar código; no obstante, en el apartado *Ejercicios* de la sección 4.2.9 se introduce al alumno en el empleo de los cuadernos de Google Colab. En la figura A.1 se observa el aspecto visual desde el navegador del inicio de este cuaderno. En esta imagen se observa a la izquierda el índice del cuaderno, compuesto por cuatro ejercicios con sus soluciones correspondientes. Estos ejercicios se muestran en forma de desplegable, y el código se puede ejecutar en el propio navegador web.

Figura A.1: Aspecto visual del cuaderno de Google Colab.

A continuación se muestra una parte del código de este cuaderno, correspondiente a los dos primeros ejercicios propuestos relacionados con la creación de gráficas.

```
1  # -*- coding: utf-8 -*-
2  """Cuaderno_Python.ipynb
3
4  Automatically generated by Colaboratory.
5
6  Original file is located at
7      https://colab.research.google.com/drive/1
           MV_PKIMXun1RKmETiMxVLcoJ6R2rMRkS
8
9  # **Ejercicio 1**
10
11 **ENUNCIADO.** Cree un histograma a partir de la siguiente lista de
       longitudes (pb) de una serie de 23 genes.
12 ```
```

```
Longitudes = [506, 177, 69, 824, 784, 243, 293, 374, 383, 527, 611,
    531, 566, 740, 482, 636, 525, 1066, 409, 590, 1110, 433, 433]
```

```

## **Solucion**
"""

import matplotlib.pyplot as plt

x = [506, 177, 69, 824, 784, 243, 293, 374, 383, 527, 611, 531,
   566, 740, 482, 636, 525, 1066, 409, 590, 1110, 433, 950]

fig, ax = plt.subplots()
ax.hist(x, color="blue", orientation="vertical", ec="black")
plt.title("Histograma", color="black", fontsize=15, fontweight="
   bold")
plt.ylabel("Frecuencia de genes", color="black", fontsize=10,
   fontweight="bold")
plt.xlabel("Longitud (pb)", color="black", fontsize=10, fontweight=
   "bold")
plt.show()

"""# **Ejercicio 2**

**ENUNCIADO.** Cree un grÃąfico de sectores con el porcentaje de A,
   T,G y C del siguiente fragmento de ADN.
```

```
fragmento = 'ATGCGCGATCGCCTAAAACGGGGTGTAGCT'
```

```

## **Solucion**
"""

import matplotlib.pyplot as plt

segmento_DNA = 'ATGCGCGATCGCCTAAAACGGGGTGTAGCT'
#Definimos cada base
citosina = 'C'
guanina = 'G'
adenina = 'A'
timina = 'T'

#Almacenamos en una variable el conteo de cada base
```

```
50 n_citosinas= segmento_DNA.count(citosina)
51 n_guaninas= segmento_DNA.count(guanina)
52 n_adenina= segmento_DNA.count(adenina)
53 n_timina= segmento_DNA.count(timina)
54
55 x=[n_citosinas, n_guaninas, n_adenina, n_timina]
56
57 #Representamos
58 nombre = ["Citosina", "Guanina", "Adenina", "Timina"]
59 fig, ax = plt.subplots()
60 ax.pie(x, labels=nombre, autopct="%1.1f%%", pctdistance=0.5)
61 plt.title("Diagrama␣sectores", color="black", fontsize=15,
      fontweight="bold")
62 plt.show()
```

A.3. Ficheros de datos

Para el seguimiento de gran parte de los apartados incluidos en la sección *Biopython*, que puede encontrarse en este trabajo en la sección 4.2.11, se proporcionan una serie de archivos en formato FASTA al alumno. A continuación se muestra el contenido de uno de los ficheros FASTA incluidos en el tutorial, en este caso utilizado para utilizar el programa informático BLAST, y que contiene la secuencia codificante del gen NAC de *Solanum lycopersicum* y de *Arabidopsis thaliana*. Su descarga es importante pero no obligatoria, ya que el alumno podría trabajar inicialmente con los ficheros FASTA que desee, aunque no es lo recomendado.

NAC-SOLANUM-ARABIDOPSIS.fasta

```
>NC_015441.3:2856617-2858593 Solanum lycopersicum cultivar Heinz 1706 chromosome 4,
SL3.0, whole genome shotgun sequence
GTCAAAGAAACTGAAACTAACACAAAGCAGGAGCAGGAGCAGCAACAAACAGAGAGAAGAAAACAGAGGA
AGATAAGAGGAAAATTTATCGAATTCGAATCGAGAGAAAAGGGGAAGTGAAGTTGCGAAGAGTGAGAATT
TCAAAGGAAATGAACAAAGGAGCAAACGGAAATCAGCAATTGGAGTTACCGGCGGGATTCAGATTCCATC
CGACAGACGACGAATTGGTGCAGCACTATCTCTGCAGGAAATGCGCCGGACAGTCGATTGCTGTATCAAT
TATAGCTGAAATTGATCTTTACAAGTTTGATCCATGGCAGTTGCCTGGTAATTTTTGACTCTTCCTCCAA
TTTTTCATAGCTTGCAAAGTCTAGGTCAAAAAAAAAATCGAAACAGATTAAGCTTCGCGTTTGATCACAG
TTTTTGGATCTGTTCCGCCATGGAATTTGAAAATCAGATCAAGTTCCTTAAAACAAGTTTTTCGACTTAT
AGCATAGCAACACAACTTCAAATCTCAAAAATCACAATTTCAAAAACTCATAGATTTCAACTTCTATATT
CAAACAGGAGCTAAATTTCGAATTTTAAAAAATTATCTTCAAATATCCGTAAAAGTTAATCTTCGAACAT
TTTTAAGTTCCTAAAACTGTTTGGTTCTTTTTCTGAACTTTATTTTTTGTTGGTAATTGCAGAGAAGGCT
TTGTACGGTGAAAAAGAGTGGTATTTTTTCTCACCAAGGGATAGAAAATATCCGAACGGTTCACGGCCGA
ACCGAGCAGCAGGAACCGGTTATTGGAAGGCAACCGGAGCTGATAAACCGGTGGGAAAACCCAAAACCTT
AGGGATAAAGAAGGCACTTGTGTTCTATGCCGGAAAAGCACCCAGAGGTATAAAAACAAATTGGATTATG
CACGAGTACCGCCTCGCCAACGTGGACCGCTCTGCTGGCAAGAACAATAACTTGAGGGTAAGTCCTTCTT
```

CAGCTTTAATTTTTTATTTTCGGAGATGAGATTCAAGATTCGAGTAATATAGTATGGTAAAATTACTGAG
TTTACGTGAACTATACTCACATCTCTAATAGTGATATCTCCTAAGATTTACAATGTTTCAAATGAAAAAT
AGTGGAAGATTTCACTTTAATTATCAACAATTAGGTGTGGCACCCATTTAACATATAGTAATAAGTAAAA
TAAAATGGTCATTATATTCTGTAACTTATGAGTGCAAAATACAATTTGTAAATTGTTTATTGATTATGTG
ATTTGACATTATTGCAGCTTGATGATTGGGTATTGTGTCGAATATACAACAAGAAAGGCACACTTGAGAA
GCATTACAATGTGGACAACAAGGAAACTACAAGCTTTGGAGAATTTGATGAAGAAATAAAACCAAAAATA
TTGCCCACACAATTAGCACCGATGCCACCACGGCCTCGATCGACACCAGCAAACGACTACTTTTATTTCG
AGTCATCAGAGTCGATGACTAGAATGCACACGACAAACTCGAGCTCTGGCTCAGAGCATGTCTTGTCGCC
ATGTGACAAGGAGGTTCAGAGCGCGCCCAAATGGGACGAAGACCACAGAAACACCCTTGATTTTCAGCTA
AACTATTTGGATGGTTTACTAAATGAACCATTTGAAACCCAAATGCAGCAGCAAATTTGCAACTTTGACC
AGTTCAACAATTTCCAAGACATGTTCCTATACATGCAAAAACCTTACTAAAATTGTATAAATTCATTGGA
TCTAAATTGAGTGTGATCCATGACATTTTCTTTGTTCTTTGGTGGTGTAGGTCAACTTTTTATTAAGTAG
TTTAGAGAAGTACAAAATGCTAGTCAAATTTGGTGGGCTACAGCACAAATGAGCCTTGATAAGCATAGCC
AAAGAGTCGTATAGAAGGGCTTATTATTATTGTAAGGTATGTAAAAACAAATGAAAATTTGTTAATATCA
AGTTATCATTCTTCAAA

>NC_003070.9:c11867154-11865229 Arabidopsis thaliana chromosome 1 sequence
GTCAAAGAAACTGAAACTAACACAAAGCAGGAGCAGGAGCAGCAACAAACAGAGAGAAGAAAACAGAGGA
GAGTCCTACCACCATTATAAATTATCTCATCGTTTGCTTTCTTTTTTTTAAGTTGTAACCATTTCCACTC
GTAATCATACAACTTCTCTACTCTTCTAGAGCAAAAACCCAAAAATATATTGCTATCTTCGTTAATGGCT
GATAATAAGGTCAATCTTTCGATTAATGGACAATCAAAAGTGCCTCCAGGTTTCAGATTCCATCCCACCG
AAGAAGAACTTCTCCATTACTATCTCCGTAAGAAAGTTAACTCTCAAAAGATCGATCTTGATGTCATTCG
TGAAGTTGATCTAAACAAGCTTGAGCCTTGGGATATTCAAGGTTAAACCCTAAATATGAAAAAAAACTCT
TTCATTTCTACGAACTCTCTATTTCTTTGTTCTTTATTTTTGAACCATTTTGGTATGCAAATGCAGAGGA
ATGTAGAATCGGTTCAACGCCACAAAACGACTGGTACTTCTTCAGCCACAAGGACAAGAAGTATCCAACC
GGGACCAGGACGAACCGGGCAACAGTCGCTGGATTCTGGAAAGCTACCGGACGTGACAAAATCATCTGCA
GTTGTGTCCGGAGAATTGGACTGAGGAAGACACTCGTGTTCTACAAAGGAAGAGCTCCTCACGGTCAGAA
ATCCGACTGGATCATGCATGAGTATCGCCTCGACGATACTCCAATGTCTAATGTAAGTATATATAGCAAT
CATCAATTTGCAGTTTAGGTTAGAATAAACAAATTACAAACTTATAGTCAAAGAATTAGCTAGATTCGTT
TCATGCTATAAAAGGTTTTGGTTCATTTAGTACGCACTAGAGCCTTCTTTTAGCATCTGCAATTAAAGTA
ATCGAACATGATTCTCTGAGTTACAATAAGATAATTTCTTATGAAATTAACACATCATATATACATTTAT
ATAAAAAGATTTTTTCACAAGGCTTAAAATAGTTTAACACGGCTCTACAAACAATACGTAGTTAACTAGT
TATGGTTGCATATCCAAAAGAAAAACCAACATAAAAGATTTTTTTTTTTGTTTCAAAATAAAAGTTGCAT
TTTTACATCATCAAGATTTAATCATCTCTTATTGCATTTCATAAGATCGATTTGTGTTCCATTATCAGGG
CTATGCTGATGTTGTTACAGAAGATCCAATGAGCTATAACGAAGAAGGTTGGGTGGTATGTCGAGTGTTC
AGGAAGAAGAACTATCAAAAGATTGACGATTGTCCTAAAATCACTCTATCTTCTTTACCTGATGACACGG
AGGAAGAGAAGGGGCCCACCTTTCACAACACTCAAAACGTTACCGGTTTAGACCATGTTCTTCTCTACAT
GGACCGTACCGGTTCTAACATTTGCATGCCCGAGAGCCAAACAACGACTCAACATCAAGATGATGTCTTA
TTCATGCAACTCCCAAGTCTTGAGACACCTAAATCCGAGAGCCCGGTCGACCAAAGTTTCCTGACTCCAA
GCAAACTCGATTTCTCTCCCGTTCAAGAGAAGATAACCGAAAGACCGGTTTGCAGCAACTGGGCTAGTCT
TGACCGGCTCGTAGCTTGGCAATTGAACAATGGTCATCATAATCCGTGTCATCGTAAGAGTTTTGATGAA
GAAGAAGAAAATGGTGATACTATGATGCAGCGATGGGATCTTCATTGGAATAATGATGATAATGTTGATC
TTTGGAGTAGTTTCACTGAGTCTTCTTCGTCTTTAGACCCACTTCTTCATTTATCTGTATGATTATCTAT
ACATACACATACACAAAATATAAAATAAAGTTTGTTAAAAAATGTCTCATGTTCTATTGATAGTACGTCT
TGATTTTGGGGTTTAACATTTGTTAATGGTGGAGGG

Apéndice B

Fichero mkdocs.yml

El fichero *mkdocs.yml* es un archivo de configuración creado automáticamente al iniciar un nuevo proyecto de Mkdocs. Este archivo alberga todas las características y configuraciones deseadas del sitio web: nombre, autor, url, índice de navegación, aspectos estéticos del tema *material*, etc.

```
1  site_name: TUTORIAL PYTHON PARA BIOQUIMICOS
2  site_author: Victor J. Fernandez
3  site_url: https://victorffdez.github.io/
       Python_Biochemistry_Tutorial
4  repo_name: victorffdez/GitHub
5  repo_url: https://github.com/Victorffdez/
       Python_Biochemistry_Tutorial
6  edit_uri: ""
7
8  nav:
9      - Inicio: index.md
10     - Introduccion a Python: 2_Introduccion_python.md
11     - PyCharm: 3_Pycharm.md
12     - Variables y operadores: 4_Variables.md
13     - Estructuras de datos: 5_EstructurasDatos.md
14     - Estructuras de control: 6_EstructurasControl.md
15     - Funciones: 7_Funciones.md
16     - Archivos y datos: 8_ArchivosyDatos.md
17     - Graficacion: 9_Graficacion.md
18     - Ejercicios complementarios: 10_Ej_complementarios.md
19     - BioPython: 11_Biopython.md
20     - Acerca de: AcercaDe.md
21
22 theme:
23     name: "material"
24     logo: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3
           /Python-logo-notext.svg/1200px-Python-logo-notext.svg.png
```

```
25      favicon: https://upload.wikimedia.org/wikipedia/commons/thumb/c
           /c3/Python-logo-notext.svg/1200px-Python-logo-notext.svg.png
26      icon:
27        repo: fontawesome/brands/github
28      palette:
29          - scheme: default
30            toggle:
31                  icon: material/toggle-switch
32                  name: Switch to dark mode
33          - scheme: slate
34            primary: red
35            accent: indigo
36            toggle:
37                  icon: material/toggle-switch-off-outline
38                  name: Switch to light mode
39      language: es
40      features:
41        - navigation.instant
42        - navigation.top
43        - search.suggest
44        - search.highlight
45        - search.share
46        -
47  plugins:
48    - search:
49        lang:
50          - es
51          - en
52  extra:
53    search:
54      language: es, en
55    social:
56      - icon: material/twitter
57        link: https://twitter.com/Fac_CienciasUCO?ref_src=twsrc%5
             Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor
58      - icon: material/school
59        link: https://uco.es/
60      - icon: fontawesome/brands/linkedin
61        link: https://www.linkedin.com/school/facultadcienciasuco/?
             originalSubdomain=es
62      - icon: fontawesome/brands/python
63        link: https://es.python.org/
64    generator: false
```

```
65
66 copyright: Copyright 2021-2022 Victor J. Fernandez. Todos los
      derechos reservados.
67 markdown_extensions:
68   - attr_list
69   - md_in_html
70   - admonition
71   - pymdownx.details
72   - pymdownx.superfences
73   - pymdownx.emoji:
74         emoji_index: !!python/name:materialx.emoji.twemoji
75         emoji_generator: !!python/name:materialx.emoji.to_svg
76   - pymdownx.critic
77   - pymdownx.caret
78   - pymdownx.keys
79   - pymdownx.mark
80   - pymdownx.tilde
81   - pymdownx.highlight:
82       anchor_linenums: true
83   - pymdownx.inlinehilite
84   - pymdownx.snippets
85   - pymdownx.superfences
86   - tables
```

Apéndice C

Fichero Markdown

A continuación se muestra el apartado *Extracción y escritura de datos* del fichero Markdown correspondiente a la sección *Biopython* del tutorial. En este archivo, se pueden observar algunos aspectos característicos de la sintaxis en Markdown comentados en el apartado 4.1.3.

````
### ***Extraccion y escritura de datos***
El archivo FASTA utilizado de ejemplo empieza de la siguiente forma

```
>HM590365.1 Viola tricolor voucher personal collection:I. Hiiesalu 74 tRNA-Leu (trnL) gene, partial sequence; chloroplast
GACTTAATTGGATTGAGCCTTGGTATGGAAACTTACTAAGTGGATAA....
....
```

Como puede comprobar, el identificador es el primer argumento en la descripcion, informacion que podemos utilizar para obtener una lista con todos los IDs:

``` py linenums="1"
from Bio import SeqIO
especies = []
for seq_record in SeqIO.parse("viola_tricolor.fasta", "fasta"):
    especies.append(seq_record.description.split()[0])
print(especies)
```

Esto te devuelve por pantalla la siguiente lista:

```
````
````

```
['HM590365.1', 'JZ084087.1', 'JZ084086.1', 'JZ084085.1','JZ084084.1', ...]
```

!!! note "Lectura de otros formatos de archivo"

    Le recomendamos que pruebe este metodo con archivos de EMBL o GenBank.

La modificacion de datos en el archivo es muy simple. Veamos como modificar por ejemplo el ID de la primera entrada:

```py linenums="1"
from Bio import SeqIO
entradas = SeqIO.parse("viola_tricolor.fasta", "fasta")
primera_entrada = next(entradas)
print(primera_entrada)

primera_entrada.id = "ID_NUEVO"
print(primera_entrada)
```