



TALLER PROGRAMACIÓN ORIENTADA A LA BIOLOGÍA

(En el marco del II CONCURSO DE BIOINFORMÁTICA EN EL AULA)

Guía de ejercicios complementarios

La bioinformática es la aplicación de métodos computacionales al análisis de datos biológicos para poder contestar numerosas preguntas. Las tecnologías computacionales permiten, entre otras cosas, el análisis en plazos cortos de gran cantidad de datos (provenientes de experimentos, bibliografía, bases de datos públicas, etc), así como la predicción de la forma o la función de las distintas unidades biológicas, o la simulación del comportamiento de sistemas biológicos complejos. Además de ser un campo de estudio dentro de la bioquímica y la biología, la bioinformática puede ser pensada como una herramienta en el aprendizaje de la biología: su objeto de trabajo son entidades biológicas (ADN, proteínas, organismos completos, sus poblaciones...), estudiadas con métodos que permiten ‘visualizar’ distintos procesos de la naturaleza. Así, la bioinformática aporta una manera clara y precisa de percibir los procesos biológicos, y acerca a los estudiantes herramientas que integran múltiples conocimientos (lógico-matemático, biológico, físico, estadístico) generando un aprendizaje significativo y envolvente.

Bienvenido!! ¿Estás listo?

Para comenzar podríamos definir algunas cosas:

¿En qué consiste una computadora?

Una computadora está formada por el Hardware (que son todas las partes o elementos físicos que la componen) y el Software (que son todas las instrucciones para el correcto funcionamiento del Hardware). El software operativo o sistema operativo proporciona una interfaz con el usuario y es el que permite al resto de los programas una interacción correcta con el Hardware.

¿Qué hacemos entonces cuando programamos?

Los lenguajes de programación actúan como un traductores entre el usuario y el equipo. Un programa intermedio, denominado *Compilador*, convierte las instrucciones dadas en un determinado lenguaje al *lenguaje máquina* (un lenguaje simple, que sólo consta de ceros y unos). Una computadora está constituida, básicamente, por un gran número de circuitos eléctricos que pueden ser activados (1) o desactivados (0). Al establecer diferentes combinaciones de prendido y apagado, se logra que el equipo realice alguna acción, por ejemplo, que muestre algo en la pantalla. Eso es la programación.

En lugar de aprender el lenguaje nativo de la máquina, se puede utilizar un *lenguaje de programación* para dar instrucciones al equipo de un modo que sea más fácil de aprender y entender. Esto significa que, como programador de Python (o cualquier otro lenguaje), no necesitas entender lo que el equipo hace o cómo lo hace, basta con que entiendas cómo funciona el lenguaje de programación.

¿Por qué es útil aprender a programar?

Tu smartphone, tu Playstation o Wii no serían muy útiles sin programas (aplicaciones, juegos, etc) para hacerlas funcionar. Cada vez que abrimos un documento para hacer un trabajo práctico para la escuela o usamos el WhatsApp para chatear con nuestros amigos estamos usando programas que interpretan lo que deseamos (un color de fuente, un formato de letra, ‘enviar un mensaje’) y se lo comunican a nuestra PC o teléfono para que ejecute nuestra orden. Aprendiendo a programar podrías hacer una gran diversidad de cosas: desde escribir tus propios juegos, aplicaciones para celular o



combinar el uso de varios programas de forma secuencial o leer millones de texto sin abrir un solo libro, hasta analizar el genoma de un organismo o un millón de estructuras proteicas y sacar conclusiones de relevancia biológica.

Entonces: ¿Qué es Python?

Es un lenguaje de programación con una forma de escritura (sintaxis) de fácil lectura. Es lo que se conoce un lenguaje de scripting, que puede ser ejecutado por partes y no necesita ser compilado en un paso aparte. Python tiene muchas características, ventajas y usos que vamos a pasar por alto en este taller, pero que puedes leer en las páginas oficiales de Python y Python Argentina. Para nosotros, por el momento, la razón más importante para elegir Python es que podés comenzar a hacer tus propios programas realmente rápido.



¿Cómo usamos Python?

Esto dependiendo de con qué dispositivo contemos. Si estás en tu **computadora**, según el sistema operativo podés instalar Python de la siguiente manera:

Windows

Podés descargar Python para Windows desde el sitio web de Python (ver QR). Después de descargar el archivo *.msi, debés ejecutarlo (hacé doble clic en el archivo) y seguir las instrucciones.

Es importante recordar la ruta (el directorio) donde se ha instalado Python. Algo para tener en cuenta: en la segunda pantalla del asistente de instalación, llamada "Customize", asegurate de ir hacia abajo y elegir la opción "Add python.exe to the Path".



Linux

Para verificar si ya lo tenés instalado (y qué versión es), podés abrir una consola y tipear el siguiente comando:

```
$ python3 --version
```

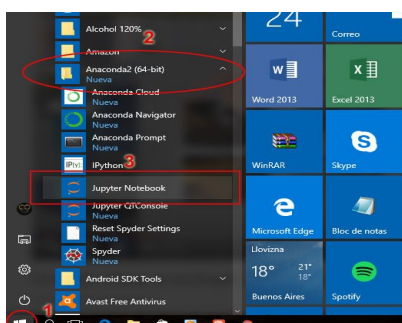
Si no tenés Python instalado o si querés una versión diferente, podés instalarlo como sigue (tipeando este comando en tu consola):

```
sudo apt-get install python3.4
```

Jupyter Notebook y Anaconda: Una interfaz amigable para correr Python

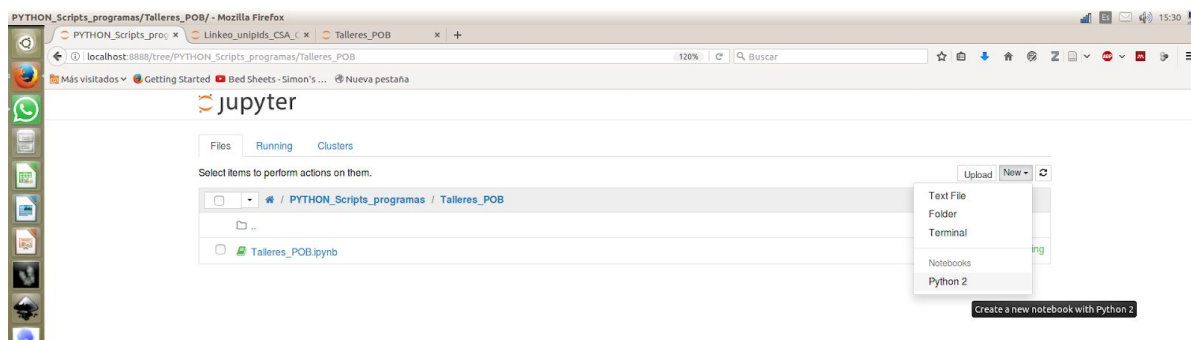
Jupyter es un intérprete de comandos para el lenguaje de programación Python, esto es un entorno interactivo (no vemos solo la terminal negra, sino que tiene botones

para navegar nuestra PC, ejecutar programas, etc.) y amigable que facilita el proceso de programación. Su instalación y uso son gratuitas y existen distribuciones para todos los sistemas operativos (<https://www.continuum.io/downloads>). Y tiene como ventajas que posee incorporadas librerías muy útiles para manipulación de datos científicos, un formato de navegación de datos amigable y las celdas de ejecución por partes de nuestro código.

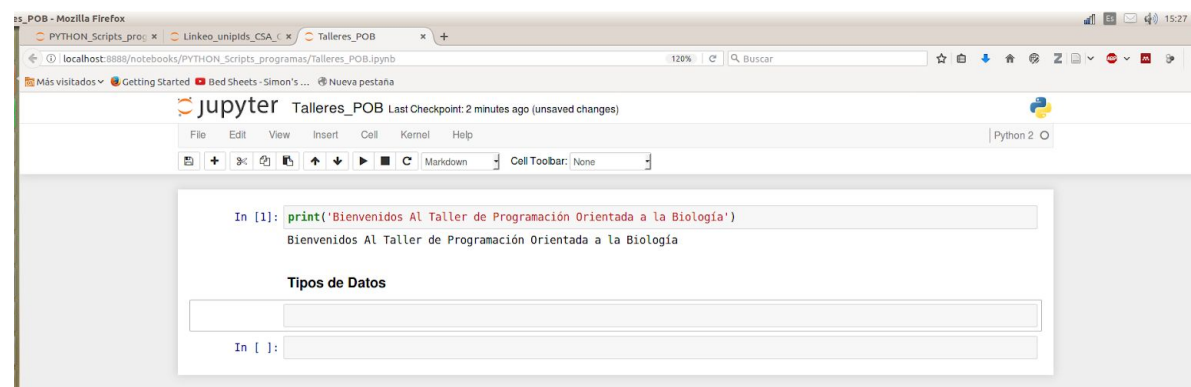


En el caso particular de windows es útil recordar para la instalación instalarlo para todos los usuarios y el path a donde se instala tiene que estar sin espacios (por ejemplo: C:\Users\Yo\Anaconda2). La interfaz se abre buscando en el inicio la carpeta Anaconda y hacés click en Jupyter Notebook. Se van a abrir una consola y una nueva pestaña en el navegador con la interfaz de Jupyter.

Como podés ver, la interfaz te permite: explorar los archivos que tengas en tu PC (botón: 'Files'), generar archivos nuevos de texto (botón: New -> Text File) o lo que se conocen como Notebooks que



son archivos de formato '.ipynb' (botón: New -> Python 2), donde se almacenan todas la información de la sesión en la que se haya trabajado. El entorno de Jupyter nos permite trabajar con estos archivos de forma interactiva: creando nuevas celdas donde ejecutar el código que queremos probar (botón:



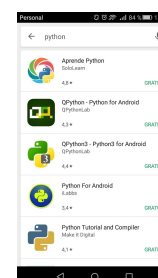
'+'), eliminando celdas que no querramos conservar en el archivo (botón: 'cut', las tijeras), o guardando todo lo escrito (botón: 'guardar', el disquete), etc. Para ejecutar cada celda con el código que hemos escrito solo tenemos que hacer SHIFT + ENTER. Aquí no veremos el prompt de Python, pero sabremos cuando está corriendo nuestra celda porque veremos al costado 'In[*]' y al finalizar la corrida de la celda se verá entre corchetes el número de celdas ejecutadas, en nuestra foto 'In[1]'.

¿Y si tenés un teléfono inteligente?

Buena noticia! Existen aplicaciones gratuitas que instalan una consola de Python en tu teléfono celular. Solo debés buscar en tu tienda 'Python' y descargarla la App:

- App para teléfonos android te recomendamos: QPython3
- App para teléfonos windows te recomendamos: Python 3.
- App para IOS: Pythoni 2.7.

Todas estas Apps son gratuitas!



¿Qué otras formas tenés para correr Python?



Bueno, existen lo que se llaman consolas en línea que te permiten correr Python como si estuviese instalado en tu PC y son completamente gratis (bueno siempre que tengas internet!), por ejemplo: repl.it (<https://repl.it/languages/python3>) y tutorialpoint.com (https://www.tutorialspoint.com/execute_python_online.php). Te recomendamos dos, pero siempre puedes buscar otras.



"Aún el camino más largo siempre comienza con el primer paso" - Lao Tse

Un primer paso para poder hacer tu primer programa es abrir la consola de Python, tu App del teléfono o consola en línea, lo que tengas a mano para arrancar!

Si estás desde tu PC la forma es:

Linux

Abrir una terminal (si no tenés creado el acceso directo, podés tipear en el inicio 'terminal' para encontrarla) y escribir:

`python`

```

usuario@usuario-MS-7623: ~
usuario@usuario-MS-7623:~$ python
Python 2.7.13 [Anaconda 2.3.0 (64-bit)] (default, Dec 20 2016, 23:09:15)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>>

```

Windows

Ir al inicio, buscar la carpeta de Anaconda. Ingresar y clickear en el ícono de Jupyter Notebook. Abrir un nuevo Notebook (New -> Python 2) y listo! Si querés trabajar desde la consola de Python sin la interfaz interactiva: buscá en el inicio Python 2.7. Hacé click en aplicación Python Prompt y ya estás listo para empezar!

Si todo está bien, vas a ver lo que se denomina 'prompt', que en el caso de Python son tres signos mayor '>>>'. El prompt aparece siempre que Python esté listo para recibir una nueva orden.

AHORA SÍ: Manos al teclado!!

Quiero ser un Pythonista

1) *La importancia de las comillas en Python.* Como vimos durante el taller, el operador matemático “+” no solo sirve para sumar dos números, sino también para concatenar palabras (*strings*). Utilizando nada más que la función `print()`, el símbolo + y la cantidad necesaria de números 7 y comillas “ ”, ¿podrías hacer programas que impriman las tres líneas siguientes? **PISTA:** como dice el título, las comillas (o la falta de ellas) es crucial.

```
14
7+7
77
```

2) *¡Similar, no es igual!* Como bien sabemos, Python distingue fácilmente entre dos palabras diferentes: ¡caracteres distintos, palabras distintas! Además, no solo podemos decidir si dos *strings* son iguales o no, si no que Python nos permite evaluar cuál de las dos palabras es la más “grande” mediante los operadores relacionales que ya conocemos: “>” y “<”. Tomando como referencia el orden alfabético, las palabras más “grandes” serían las que aparecen más tarde en nuestro alfabeto. Python también es capaz de distinguir sutilezas no tan sutiles, como diferencias en mayúsculas y minúsculas. Palabras con iguales caracteres pueden ser distintas para Python si contienen mayúsculas, por ejemplo ‘Genes’ o ‘genes’. Teniendo en cuenta todo esto:

- Escribí un programa que indique si “a” es mayor, menor o igual que “z”.
- Escribí un programa que indique si “a” es mayor, menor o igual que “A”.
- Escribí un programa que indique si “a” es mayor, menor o igual que “Z”.
- Escribí un programa que indique si “A” es mayor, menor o igual que “Z”.
- Sobre la base de los resultados anteriores y de acuerdo a Python, ¿qué alfabeto de Python está antes, el de las minúsculas o el de las mayúsculas?

3) *Vayamos por partes...* Hemos visto que las listas de Python pueden “cortarse en fetas” utilizando los corchetes: por ejemplo, `lista[a:b]` extrae todos los elementos de “lista” comprendidos entre las posiciones “a” y “b-1”. En el siguiente ejemplo usamos las funciones `list()` y `range()` para definir una lista que contiene los números del 0 al 9, y luego imprimimos sólo sus primeros cinco elementos:

```
lista_numeros = list(range(0,10))
print(lista_numeros[0:5])
```

Escribí un programa que genere e imprima una lista de todos los números pares entre 0 y 30 inclusive. Podés hacerlo de distintas formas: ¿cuántos programas distintos se te ocurren? **PISTA:** al “cortar en fetas”, podés definir el “salto” entre una feta y otra definiendo el parámetro “c” en `list[a:b:c]`.

4) *Te comento algo.* En Python tenemos la posibilidad de incluir texto que, aunque esté escrito en el programa, no deba ejecutarse. Esto se logra empezando la línea con el símbolo # (el *hash* o numeral). Estas líneas se llaman “comentarios” y se utilizan para incluir en el programa algunas aclaraciones acerca del código útiles para nosotros, por ejemplo:

```
# definir la variable 'nombre' e imprimirla
nombre = "Ana"
print(nombre)
```

Sobre la base del ejemplo anterior, indicá cuál es el efecto (y qué lo causa) de agregar un símbolo # al principio de las tres líneas presentadas debajo, de a una por vez:

```
# imprime del 0 al 9
for i in range(0,10):
    print(i)
```


La Bioinformática es lo mío

1) Dentro de la célula, el ADN guarda la información necesaria para construir las proteínas. En este camino, el mensaje genético es transportado desde el ADN por otra molécula, el ARN, muy similar al ADN pero con el nucleótido U en vez de T. En el ARN, cada triplete o grupo de tres nucleótidos/letras se denomina codón, y cada aminoácido de las proteínas está codificado por uno o varios codones. Por ejemplo, el codón 'AUG' codifica para el aminoácido Metionina, el codón 'AAA' para Lisina, el codón¹ 'CUA' para Leucina, etc. Tomemos por ejemplo la siguiente codificación², una versión simplificada del código genético universal²:

Aminoácido	3 letras	1 letra	Codón ARN
Alanina	Ala	A	GCU
Arginina	Arg	R	AGA
Asparagina	Asn	N	AAU
Aspartato	Asp	D	GAU
Cisteína	Cis	C	UGC
Fenilalanina	Phe	F	UUU
Glicina	Gly	G	GGC
Glutamato	Glu	E	GAA
Glutamina	Gln	Q	CAG
Histidina	His	H	CAU
Isoleucina	Ile	I	AUU
Leucina	Leu	L	UUA
Lisina	Lys	K	AAA
Prolina	Pro	P	CCU
Serina	Ser	S	UCU
Tirosina	Tyr	Y	UAU
Treonina	Thr	T	ACU
Triptófano	Trp	W	UGG
Valina	Val	V	GUA
Metionina/Codón de Inicio	Met/Start	M	AUG
Codón de terminación	Stop		UAA

Como una forma de "acortar" la cantidad de letras que se usan para simbolizar un aminoácido, se establecieron por convención las **nomenclaturas de 3 letras y de 1 letra**. Entonces, por ejemplo, la forma más corta de simbolizar Asparagina puede ser usando "Asn" o, más simple aún, solo "N".

- a) ¿Podrías escribir un programa que imprima una cadena de ARN codificante para el siguiente péptido (cadena corta de aminoácidos). **PISTA: usá bucles, condiciones y operadores.**

Sec1:

'AGEKGKKIFVQKCSQCHTVCSQCHTVEKGGKHKTGPNEKGKKIFVQKCSQCHTVLHGLFGRKTGQA'

- b) ¿Podrías hacer que tu programa cuente el contenido de G y C (en términos de porcentaje, respecto al largo total de la secuencia) sobre la secuencia de ARN generada?

2) En muchos de los genes codificados en el ADN existe un motivo recurrente ubicado antes de la secuencia codificante del gen. Se trata de una palabra específica para la unión de ARN Polimerasa II,

¹ Codón: La información genética transportada por el ARN consiste de repeticiones de cuatro letras (A, C, G y U), que corresponden a sus bases nitrogenadas, y están dispuestas en una determinada secuencia. Estas letras pueden agruparse funcionalmente de tres en tres, en 64 formas distintas. Cada uno de estos grupos de tres letras se llama codón. Cada codón codifica un aminoácido o un símbolo de puntuación (comienzo o parada de la construcción de la proteína correspondiente).

² El código genético es el conjunto de reglas que define cómo se traduce una secuencia de nucleótidos en el ARN a una secuencia de aminoácidos en una proteína. Este código define la relación entre cada secuencia de tres nucleótidos, llamada codón, y cada aminoácido; y se dice que es universal ya que el mismo triplete en diferentes especies codifica para el mismo aminoácido. La principal excepción a la universalidad es el código genético mitocondrial.

la proteína encargada de copiar el ADN a un ARN mensajero, y su secuencia es 'TATAAA'. Para los siguientes fragmentos de ADN, ¿podrías identificar si poseen la región de unión a la polimerasa³?

Sec2:

```
'GTTATAATATTGCTAAAATTATTCAGAGTAATATTGTGGATTAAGCCACAATAAGATTTATAATCTTAAATGATGGGACTACCATCCTT
ACTCTCTCCATTTCAAGGCTGACGATAAGGAGACCTGCTTTGCCGAGGAGGTACTACAGTTCTCTTACAAACAATTGTCTTACAAAATGA
ATAAACAGCACTTTGTTTTATCTCTGCTTTTAATATGTCCAGTATTCATTTTTGCATGTTTGGTTAGGCTAGGGCTTAGGGATTATA
TATCAAAGGAGGCTTTGTACATGTGGGACAGGGATCTTATTTAGATTTATATATCAAAGGAGGCTTTGTACATGTGGGACAGGGATCTTA
TTTTACAAACAATTGTCTTACAAAATGAATAAACAGCACTTTGTTTTATCTCTGCTCTATTGTGCCATACTGTTGAATGTTTATAATG
CATGTTCTGTTTCCAAATTTTCATGAAATCAAACATTAATTTATTTAAACATTTACTTGAAATGTTTACAAACAATTGTCTTACAAAATGA
ATAAACAGCACTTTGTTTTATCTCTGCTTTTAATATGTCCAGTATTCATTTTTGCATGTTTGGTTAGGCTAGGGCTTAGGGATTATA
TATCAAAGGAGGCTTTGTACATGTGGGACAGGGATCTTATTTAGATTTATATATCAAAGGAGGCT '
```

Sec3:

```
'ACTTTGTTTTATCTCTGCTCTATTGTGCCATACTGTGAATGTTTATACCCTACATGGTGCATGTTCTGTTTCCAAATTTTCATGAAAT
CAAAACATTAATTTATTTAAACATTTACTTGAAATGTTTACAAACAATTGTCTTACAAAATGAATAAACAGCACTTTGTTTTATCTCT
GCTTTTAATATGTCCAGTATTCATTTTTGCATGTTTGGTTAGGCTAGGGCTTAGGGATTTATATATCAAAGGAGGCTTTGTACATGTGGGA
CAGGGATCTTATTTTAGATTTATATATCAAAGGAGGCT '
```

3) Se denomina mutación puntual a los cambios que alteran la secuencia de nucleótidos del ADN. Aún cuando involucran un solo sitio o nucleótido, las mutaciones en la secuencia del ADN (y, por consiguiente, en el ARN generado a partir de él) pueden llevar a la sustitución de aminoácidos en las proteínas resultantes. Por ejemplo, imaginemos que una mutación produce el cambio de un nucleótido 'U' por 'A'. Esto daría como resultado el intercambio de un codón codificante para Leucina ('UUA') por un codón de terminación ('UAA'). De este modo, una mutación puntual puede tener como resultado una proteína más corta y no funcional, si se introduce tempranamente un codón de terminación.

Tomando nuevamente la codificación simplificada del ejercicio 1, ¿podrías construir un programa que nos diga qué longitud tendrán los péptidos que se generarán a partir de los genes especificados a continuación, si sobre ellos se produce la mutación puntual mencionada antes para el codón de Leucina?

Sec4:

```
'AAUCAUGUAGUAGGCUUUUUUUUAGAUAUGCU '
```

Sec5:

```
'GCUUAUCCUGCUUUGGCUCUGGCGUAUUAACUGGC '
```

Sec6:

```
'GCUUUAUAUCCUGCUUUGGCUCUGGCGUAUUAACU '
```

4) Existen mutaciones típicamente asociadas a ciertas proteínas. Imaginemos que trabajamos con una proteína para la que cierta mutación puntual se asocia a una enfermedad inmunológica. En dicha

³ ARN polimerasa II es una proteína de eucariotas que cataliza la transcripción del ADN a precursores de ARN mensajero, microARNs y otros tipos de ácido ribonucleico.

patología los pacientes presentan intercambiada la Histidina por una Lisina en el fragmento 'HTVCSQ'. ¿Podrías construir un programa que identifique cuáles de los siguientes pacientes poseen la enfermedad? **PISTA: el resultado puede estar en forma de lista.**

Paciente1:

'AGEKGKKIFVQKCSQCLTVCSQCHTVEKGGKHKTGPNEKGKKIFVQKCSQCHTVLHGLFGRKTGQA'

Paciente2:

'AGAKGKKIFVQKCSQCLTVCSQCHTVEKGGKHKTGPNEKGKKIVVQKCSQCLTVLLGLFGRKTGQA'

Paciente3:

'AGEKGKKIFVQKCSQCLTVCSQCHTVEKGGKHKTGPNEKGKKIFVQTAKCSQCHTVLLAKTGHLQA'

Paciente4:

'AGAKGKKIFVQKCSQCHTVCSQCHTVEKGGKHKTGPNEKGKKIVVQKCSQCLTVLLGLFGRKTGQA'

Paciente5:

'AGAKGKKIFVQKCSQCNPLCSQCHTVEKGGKHKTGPNEKGKKIVVQKCSQDETVLLGLFGRKTGQA'

5) Como resultado de la evolución biológica, y dado que todos los organismos provenimos de ancestros comunes, existen genes en distintos organismos que guardan cierto grado de similitud (es decir, comparten total o parcialmente la secuencia) y que codifican para proteínas similares. Por ejemplo, tomemos estos fragmentos de secuencias de Citocromo C (una proteína involucrada en la respiración), pertenecientes a organismos distintos:

CitC_humano:

'AGDVEKGKKIFIMKCSQCHTVEKGGKHKTGPNLHGLFGRKTGQA'

CitC_gorila:

'AGDVEKGKKIFIMKCSQCHTVEKGGKHKTGPNLHGLFGRKTGQA'

CitC_pollo:

'AGDIEKGKKIFVQKCSQCHTVEKGGKHKTGPNLHGLFGRKTGQA'

- a) ¿Podés construir un programa sencillo que compare las secuencias y diga si son o no iguales?
- b) Supongamos que esas secuencias están alineadas⁴, de forma tal que el primer aminoácido ocupa la posición número 1 en todas las secuencias, el segundo aminoácido ocupa la posición 2 en todas, etc. ¿Cuál sería la posición menos conservada?

⁴ Un alineamiento de secuencias es una forma de representar y comparar dos o más secuencias de ADN, ARN o de proteínas, para resaltar sus semejanzas y diferencias. Consiste en disponer las secuencias en líneas consecutivas, insertando espacios en cada una de ellas para que las zonas equivalentes queden alineadas verticalmente. Así, se asume que los aminoácidos o nucleótidos que quedan en la misma columna son equivalentes entre sí. Si dos secuencias en un alineamiento comparten un ancestro común, las diferencias en una posición dada pueden interpretarse como mutaciones puntuales (sustituciones), y los huecos en alguna de las secuencias, como fragmentos que se ganaron o perdieron durante su evolución.

6) Dada la siguiente lista de secuencia ¿podrías construir un programa que detecte si se trata de una secuencia de ADN, ARN o proteína e imprima en pantalla de qué tipo de molécula se trata?

Sec7:

'AGDVEKGGKIFIMK'

Sec8:

'ATTGGACGTAATGC'

Sec9:

'AGGUACCGAUCCA'

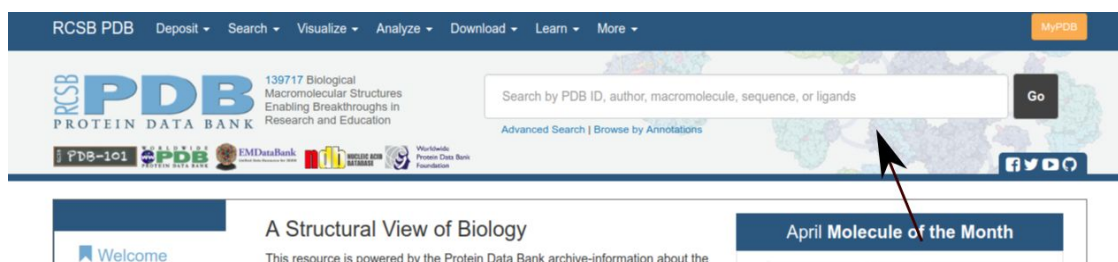
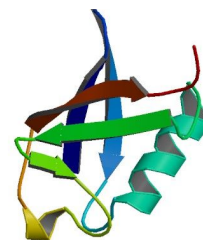
7) ¿Podrías mejorar el programa del punto anterior haciendo que sea capaz de tomar secuencias que ingresa el usuario? Recordá que no siempre el usuario utiliza mayúsculas, pero tu programa debe ser capaz de procesar tanto minúsculas como mayúsculas.

Uso y desusos de las Bases de Datos Biológicas

Hemos usado durante los talleres mucha información biológica, como secuencias de ADN, de ARN y de proteínas. ¿De dónde viene esa información? En términos generales los científicos de todo el mundo desarrollan diversos conocimientos relacionados con los seres vivos. Estos conocimientos se obtienen a base de observaciones y experimentación. Los datos y conclusiones obtenidos son compartidos entre científicos de forma organizada, ya sea por medio de publicaciones en revistas super-archi-nerds o, por ejemplo, a través de Bases de datos disponibles en internet. Una base de datos (BD) es una colección estructurada de datos; en particular, una base de datos biológica es una colección de información relacionada con seres vivos. Estos datos provienen de experimentos científicos, literatura publicada, análisis computacional, etc. La información contenida en bases de datos biológicas puede incluir, por ejemplo: funciones, estructura y localización de proteínas o genes, efectos clínicos de mutaciones, así como similitudes de secuencias o distancias evolutivas, etc. Entre las bases de datos más utilizadas por científicos de todo el mundo, bioinformáticos o no, se encuentran GenBank (colección de todas las secuencias biológicas estudiadas) y PDB (que guarda la información estructural disponible acerca de ácidos nucleicos y proteínas).

8) La ubiquitina es una proteína pequeña que ha sido encontrada en casi todas las células eucariotas (de allí viene su nombre: *ubiquo* significa omnipresente). Esta proteína es la encargada de la marcación química de las proteínas que ya no son necesarias, para que otras proteínas las reconozcan y las destruyan. (Para reflexionar: ¿Por qué una célula querría destruir sus propias proteínas?). Te invitamos a descubrir un poco más acerca de ella.

- a) Desde tu celular/computadora ingresá al sitio: <https://www.rcsb.org/>. Esta página web corresponde a una de las bases de datos más utilizadas en la bioinformática: PDB. En esta base de datos se encuentran almacenadas todas las estructuras de macromoléculas biológicas obtenidas hasta el momento, como la presentada aquí a la derecha. Las estructuras se almacenan en forma de archivos que contienen las coordenadas en el espacio, en los ejes X, Z e Y, de todos los átomos de una molécula dada. Estas coordenadas pueden ser interpretadas por algunos programas gráficos para mostrar de forma tridimensional como se vería, por ejemplo, una proteína en una célula o en solución. En el recuadro marcado por la flecha en la imagen siguiente, ingresá el siguiente código: **1UBQ**.



- b) Se va a cargar una página con todos los datos de la ubiquitina. ¡Exploralas, a ver de qué se trata!

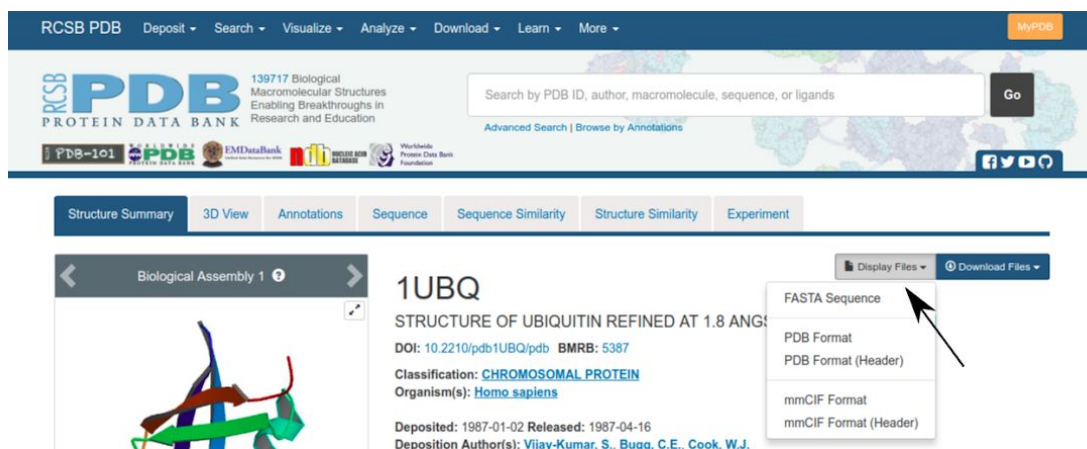
1UBQ
STRUCTURE OF UBIQUITIN REFINED AT 1.8 ANGSTROMS RESOLUTION
DOI: 10.2210/pdb1UBQ/pdb BMRB: 5387
Classification: **CHROMOSOMAL PROTEIN**
Organism(s): [Homo sapiens](#)
Deposited: 1987-01-02 Released: 1987-04-16
Deposition Author(s): [Vijay-Kumar, S.](#), [Bugg, C.E.](#), [Cook, W.J.](#)

Experimental Data Snapshot
Method: X-RAY DIFFRACTION
Resolution: 1.8 Å

wwPDB Validation

Metric	Percentile Ranks	Value
Clashscore	11	11
Ramachandran outliers	0	0
Sidechain outliers	8.8%	8.8%
RSZR outliers	5.3%	5.3%

- c) Hací click en el recuadro 'Display files'. Se va a desplegar un pequeño menú en el que vas a seleccionar 'FASTA sequence'.



- d) Luego de hacer click ahí, se va a abrir una ventana emergente que contiene la información indicada en la siguiente imagen. Esas letras son la secuencia en formato FASTA (un formato de archivo que se usa para datos biológicos). La primera línea empieza con el símbolo mayor (>) y contiene información de la proteína, como por ejemplo su nombre. La segunda línea contiene la secuencia propiamente dicha.

```
>1UBQ : A | PDBID | CHAIN | SEQUENCE
MQIFVKLTGKTTITLEVEPSDTIENVKAKIQDKEGIPPDQQRLLFAGKQLEDGRTLSDYNIQKESTLHLVLRGG
```

Declará una variable en la consola de Python, con el nombre que quieras, y almacená el contenido de la secuencia de ubiquitina en dicha variable. Ésta será una variable del tipo *string*.

- e) ¿Podrías construir un programa que calcule el porcentaje de Leucina (L) en dicha proteína? ¿Y lo mismo, pero de Treonina (T)?
- f) Ahora mejoremos el programa para que sea más “amigable con el usuario”, haciéndole las siguientes mejoras:
- Hacé que el programa le pregunte al usuario qué secuencia desea analizar.
 - Hacé que el programa consulte al usuario sobre qué aminoácido quiere calcular el porcentaje de abundancia en la proteína.
 - Hacé que el programa arroje luego la longitud de la secuencia y el porcentaje del aminoácido deseado en dicha proteína.

Nos veremos otra vez...

Hasta acá llegamos con esta guía. Les agradecemos su interés y los invitamos a acercarse a nuestro Grupo y visitar nuestra página siempre que deseen. ¡Esperamos su participación en el próximo Concurso de Bioinformática para Escuelas Secundarias!

¡Gracias totales!

Ante cualquier duda podés contactar por correo electrónico a Ana Julia Velez Rueda (anavelezrueda@gmail.com), coordinadora del proyecto, o con cualquiera de los docentes de nuestros talleres. Encontranos también a través de la página de nuestro grupo, el SBG o Grupo de Bioinformática Estructural de la Universidad Nacional de Quilmes: <http://ufq.unq.edu.ar/sbg/education/index.html>.