

Python. Trabajo final

Causas de muerte en España

Autor: Cristóbal Pareja Flores

Actualizado: 16 de noviembre de 2020

# NOTA PRELIMINAR

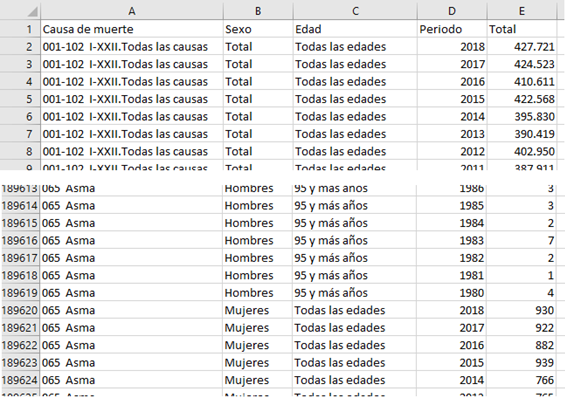
Antes de desarrollar el proyecto que se propongo a continuación, te recerdo la importancia de leer las instrucciones de entrega.

# CAUSAS DE MUERTE EN ESPAÑA

En el INE se puede encontrar un sinfín de datos abiertos sobre distintos aspectos de nuestro país, como son el mercado laboral, el nivel y las condiciones de vida, la demografía y la población o la ciencia y la tecnología. Ahora nos centramos en las defunciones según la causa de la muerte, y nos basamos en primer lugar en la información disponible en la página siguiente:

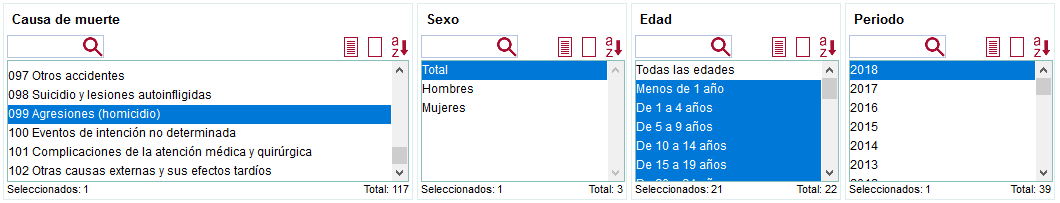
<https://www.ine.es/jaxiT3/Tabla.htm?t=7947>

El archivo completo de datos se puede descargar libremente. Se trata de una tabla de más de 300.000 registros. Para realizar una primera inspección visuals, seleccionamos una parte de la infomación, y vemos lo siguiente:

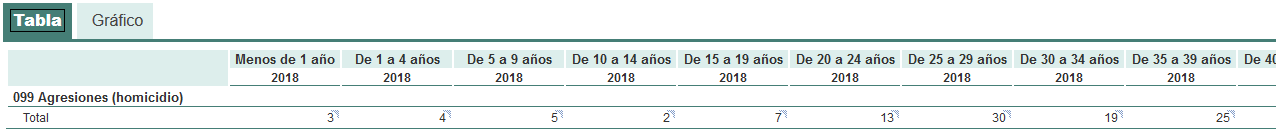


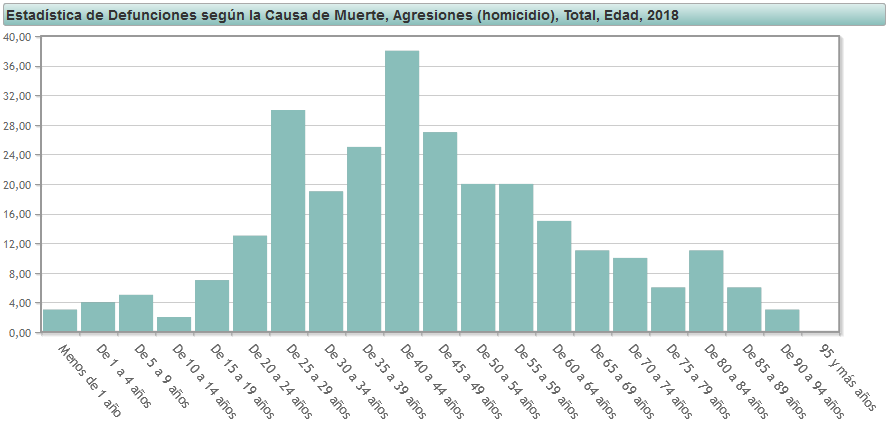
*Figura 1. Número de muertes en España, según su causa*

A fin de familiarizarnos un poco con esta infomración, solicitamos en la página web que nos muestre las muertes por homicidio (099), de personas de todas las edades sin distinción de sexo, en 2018:



Obtenemos la siguiente tabla y gráfico:





En este proyecto, queremos manejar estos datos para sacar nuestras propias conclusiones sobre distintos aspectos en relación con las causas de defunción en España.

1. **Lectura básica del archivo csv** **[1,5 puntos]**

Empezamos por descargar la tabla csv con los datos de las páginas del INE. A manera de introducción, te pido una función que carga únicamente unas pocas líneas de dicha tabla (excluida la primera), convirtiendo en enteros el año y la cantidad. Por ejemplo, la llamada siguiente:

|  |
| --- |
| **mis\_datos = cargar\_lineas("ine\_mortalidad\_espanna.csv", 7, 10)**  **for dato in mis\_datos:**  **print(dato)** |

Carga las líneas 7 a 10, del archivo "ine\_mortalidad\_espanna.csv", excluida la primera línea, y generará la siguiente salida:

|  |
| --- |
| **['001-102 I-XXII.Todas las causas', 'Total', 'Todas las edades', 2012, 402950]**  **['001-102 I-XXII.Todas las causas', 'Total', 'Todas las edades', 2011, 387911]**  **['001-102 I-XXII.Todas las causas', 'Total', 'Todas las edades', 2010, 382047]**  **['001-102 I-XXII.Todas las causas', 'Total', 'Todas las edades', 2009, 384933]** |

Lo que te pido es una función que trabaje, exactamente, como la del ejemplo anterior.

1. **Dos funciones auxiliares [2 puntos]**

En este apartado vamos a desarrollar dos funciones auxiliares, y que van a servirte en otros apartados. Son muy sencillas, pero tu solución ha de ser limpia y clara.

b.1) [1 punto] Una primera función convertirá un número romano en un entero.

Las pruebas de funcionamiento te darán la descripción del comportamiento que se espera. Por ejemplo, la llamada siguiente:

|  |
| --- |
| **print([(r, romano\_a\_entero(r)) \**  **for r in ["I", "IV", "XIV", "XXXIX", "VL", "LXIV", "MCDXCII"]])** |

generará la siguiente salida:

|  |
| --- |
| **[('I', 1), ('IV', 4), ('XIV', 14), ('XXXIX', 39), ('VL', 45), ('LXIV', 64), ('MCDXCII', 1492)]** |

Observa que en un número romano, e último símbolo suma siempre, pero los demás suma o restan según sea el siguiente de menor valor o de mayor.

b.2) [1 punto] En segundo lugar, mirando la tabla de datos, puedes observar que, en algunas líneas, se agrupan las causas de muerte por categorías. Estas líneas empiezan por tres dígitos, seguidos por un guion y otros tres dígitos:

|  |
| --- |
| **001-008 I.Enfermedades infecciosas y parasitarias Hombres De 65 a 69 años 2001 305** |

Pero cuidado, también hay algunas líneas que totalizan **todas** las causas:

|  |
| --- |
| **001-102 I-XXII.Todas las causas Mujeres De 10 a 14 años 1983 308** |

Nos interesan las primeras, pero no las segundas.

Te pido una función que dice si una línea es de la primera categoría pero no de la segunda, o sea, si cumple las siguientes condiciones:

* Empieza por tres dígitos, seguidos por un guion y otros tres dígitos.
* No totaliza **todas** las causas de fallecimiento.

La primera circunstancia (lo de los dígitos) se puede comprobar usando una expresión regular. Te popongo que lo hagas así para practicar usando esta librería, siquiera básicamente. Si lo haces sin usar la librearía de expresiones regulares, este apartado se puntúa como 0.75; si usas esta librería, 1 punto.

También en este caso, las pruebas de funcionamiento te servirán para concretar comportamiento que se espera. Por ejemplo, las instrucciones siguientes

|  |
| --- |
| **lista\_de\_causas = [**  **"009-041 II.Tumores",**  **"009 Tumor maligno del esófago",**  **"001-102 I-XXII.Todas las causas",**  **"077-080 XIV.Enfermedades del sistema genitourinario",**  **"082 XVI.Afecciones originadas en el periodo perinatal",**  **"050-052 VI-VIII.Enfermedades del sistema nervioso y de los órganos de los sentidos"**  **]**  **for causa in lista\_de\_causas:**  **print(causa.ljust(60), "\t", es\_grupo\_y\_no\_total(causa))** |

generarán la siguiente salida:

|  |
| --- |
| **009-041 II.Tumores True**  **009 Tumor maligno del esófago False**  **001-102 I-XXII.Todas las causas False**  **077-080 XIV.Enfermedades del sistema genitourinario True**  **082 XVI.Afecciones originadas en el periodo perinatal False**  **050-052 VI-VIII.Enfermedades del sistema nervioso y de los órganos de los sentidos True** |

También en este apartado b), ambas funciones han de trabajar, exactamente, como las que te doy en los ejemplos anteriores.

1. **Carga selectiva de datos de grupos de enfermedades [1,5 puntos]**

La tabla que proporciona el INE es demasiado grande: tiene más de 300.000 registros. Preferimos ahora trabajar con grupos de enfermedades:

* 001-008 I. Enfermedades infecciosas y parasitarias
* 009-041 II.Tumores
* 042-043 III. Enfermedades de la sangre y de los …
* Etcétera.

Nos interesan las filas que describen grupos unidos (y se distinguen por tener, en su primera columna, tres cifras, un guion y otras tres cifras) pero descartando las líneas con datos referidos a todas las causas. Por ejemplo, las instrucciones siguientes:

|  |
| --- |
| **datos, num\_lin\_descartadas = cargar\_datos("ine\_mortalidad\_espanna.csv")**  **print(len(datos), num\_lin\_descartadas)**  **for i in [13000, 34, 1001, 20000, 25000]:**  **print(datos[i])** |

generarán la siguiente salida:

|  |
| --- |
| **36036 265122**  **('001-008 I', 'I', 1, 'Enfermedades infecciosas y parasitarias', 1984, 3232)**  **('001-008 I', 'I', 1, 'Enfermedades infecciosas y parasitarias', 1992, 16)**  **('050-052 VI-VIII', 'VI', 6, 'Enfermedades del sistema nervioso y de los órganos de los sentidos', 2005, 19)**  **('062-067 X', 'X', 10, 'Enfermedades del sistema respiratorio', 1986, 31)**  **('074-076 XIII', 'XIII', 13, 'Enfermedades del sistema osteomuscular y del tejido conjuntivo', 2017, 0)** |

Como vemos, los números romanos han sido traducidos a enteros, y en el caso de varios grupos. Obviamente, este apartado se apoya en distintas piezas previas: la función que extrae un número romano entre “I” y “XX” de una cadena de caracteres, una función que convierte un numeral romano de los mencionados en el entero correspondiente, etc.

1. **Tablas de datos por grupos [1,5 puntos]**

Las funciones anteriores están bien para inspeccionar los datos, pero ahora deseamos almacenar el resultado en un par de tablas para su procesamiento posterior. La primera de ellas será un diccionario con los nombres de los grupos de enfermedades:

1 : Enfermedades infecciosas y parasitarias

2 : Tumores

3 : Enfermedades de la sangre y de los órganos hematopoyéticos, y ciertos trastornos que afectan al mecanismo de la inmunidad

…: Etcétera.

La segunda tabla –otro diccionario– tendrá la siguiente estructura:

* La clave será una cuaterna: el número de la causa de la muerte según la descripción anterior, el sexo, el rango de edad y el año, como un entero
* El valor asociado será el número de fallecimientos de dicha situación y año.

Por ejempo, las columnas 1 y 237.755 de la tabla dada generarán los dos registros siguientes en este diccionario:

(1, “Total”, “Todas”, 2018) : 427721

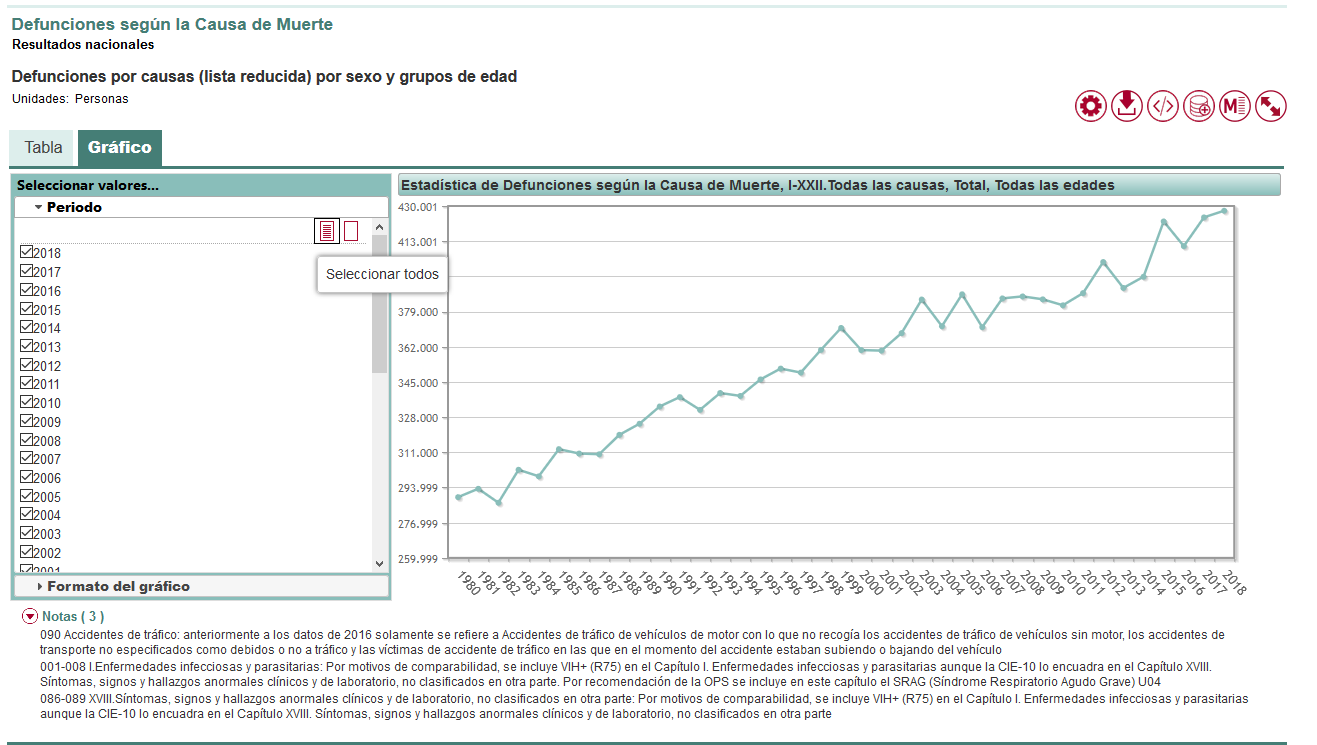
(15, “Hombres”, (1, 4), 2009) : 0

Donde “Total” se refiere al total de ambos sexos 2 y “Todas” se refiere a todas las edades, mientras que el par (1, 4) alude a las personas entre 1 y 4 años.

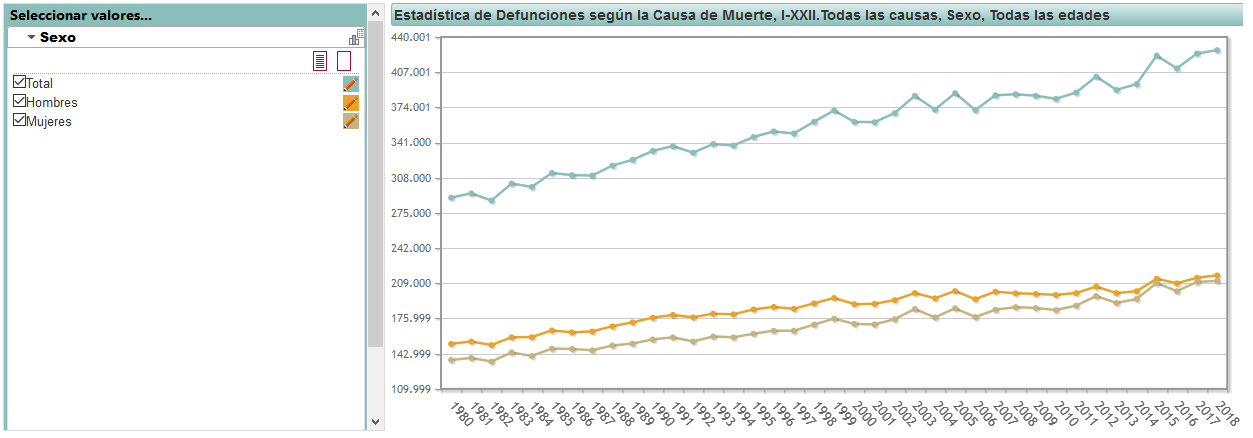
Te pido que diseñes sendas funciones que creen estas tablas. Inventa tú mismo un par de tests o tres para mostrar que las tablas se leen y crean correctamente.

1. **Tabla y gráfico de fallecimientos por todas las causas [1,5 puntos]**

Seleccionando en las páginas del INE la serie correspondiente a todas las causas, ambos sexos y de todas las edades, tenemos el número de fallecidos en España cada año. El INE nos muestra la serie de datos y la gráfica siguiente:



Diseña una función que da una gráfica como la anterior y otra similar a la siguiente, que compara la mortalidad de los dos sexos:



1. **Población española [1 punto]**

Para el siguiente apartado, necesitamos descargar, también del INE, una tabla con la población española en cada año, por sexo ("Todas las edades", "Hombres", "Mujeres") y edad ("Menos de 1 año", "De 1 a 4 años", ..., "95 y más años").

Una vez descargada o construida esta tabla, te pido una función que genere un dataframe con los datos de mortalidad que ya tienes, pero ampliados con dos columnas: una correspondiente a la población y otra, la más interesante, con la tasa de mortalidad correspondiente a cada causa, sexo, edad y año.

1. **Tasa de mortalidad [0,25 puntos]**

Con los datos del apartado anterior, deseamos construir la serie de tasas de mortalidad, en un año dado, según una causa, a medida que avanza la edad, junto con el gráfico correspondiente, donde las abcisas son los grupos de edad y las ordenadas, los pares de las distintas tasas de mortalidad de hombres y mujeres.

1. **Causas de muerte más importantes en cada edad [0,25 puntos]**

Planteamos este apartado en tres pasos (puntuados sobre 0.25 cada uno):

h.1) En primer lugar, necesitamos definir una función que, dado un año, un rango de edades y una lista de causas de muerte, genere un diagrama de sectores (tarta), para poder ver la incidencia relativa de dichas causas de muerte en ese rango de edad, ese año.

h.2) Para un año y un rango de edades, una función podría darnos las (por defecto, cinco) causas de muerte más importantes para ese año y rango de edades.

h.3) ara terminar, necesitaríamos diseñar una función con un único parámetro, un año, o quizá también el número de causas de muerte que deseamos estudiar, y genere una colección de diagramas de sectores (tarta), uno por cada grupo de edad, con las cinco causas de muerte de mayor tasa, en cada juno de dichos grupos de edad, ese año.

1. **Número de fallecimientos por cada causa (I) [0,25 puntos]**

Deseamos saber el total de muertes en un año dado por cada causa. Esto se puede calcular de varias maneras. Te pido ahora que resuelvas este problema con un programa iterativo, que recorre líneas de nuestro archivo de datos, ignora las líneas correspondientes a "Todas las causas", las de los "Hombres" y las de las "Mujeres", quedándose únicamente con las de Sexo = "Total", y totaliza el número de muertes por cada causa, así como el año mínimo y máximo tenido en cuenta.

1. **Número de fallecimientos por cada causa (II) [0,25 puntos]**

Te pido ahora que resuelvas el ejercicio anterior con la técnica map-reduce.

# APÉNDICE A. DESCRIPCIÓN DE LA ENTREGA

La entrega consistirá en una carpeta comprimida, identificada con los apellidos y nombre del estudiante (por ejemplo, “ParejaFloresCristobal”) y sin tildes ni eñes. En ella una carpeta por cada problema planteado, que contendrá los siguientes archivos:

* Un archivo ipynb con la solución de los apartados resueltos, explicados adecuadamente, con excepción del apartado10 (map-reduce). Cada apartado tendrá su rótulo, en modo markdown, con el enunciado, la solución, unos pocos ejemplares de prueba bien elegidos que muestren el funcionamiento de dicho apartado, y una explicación de lo logrado y de lo no logrado. Algún apartado se puede resolver en varios pasos con distintas funciones, y en tal caso será necesario ilustrar el funcionamiento de cada una de ellas con tests apropiados.

Al final de este archivo, se incluirá el apartado de documentación que se describe más adelante.

Este archivo se entregará también en formato pdf. En el archivo ipynb (y en el pdf correspondiente) se han de poder ver las comprobaciones parciales de cada uno de los apartados.

* Para el problema de map-reduce, el programa o los programas necesarios en Python, ejecutables desde la línea de comandos, que implementan las tareas realizadas.

Para este programa, necesitaré un archivo llamado “instrucciones\_de\_uso.txt” que describa la invocación que se debe hacer, desde la línea de comandos, del programa con map-reduce. En todo caso, el uso incorrecto de este programa deberá disparar el correspondiente aviso con las instrucciones necesarias para su correcto funcionamiento.

* Los archivos de datos completos, de manera que permitan ilustrar todas las características de las soluciones aportadas.
* Un pequeño archivo llamado “documentaciónProyecto.{docx, pdf, txt}” que contenga la información siguiente:
* Si procede, las decisiones de diseño generales o comentarios adicionales sobre el trabajo entregado.
* La ayuda recibida o fragmentos de código tomados de otro estudiante o de cualquier otra fuente.
* Una pequeña ficha de autoevaluación, es decir, el nombre y apellidos del autor y la relación de partes de la práctica logradas, junto con la nota estimada por el estudiante en cada una de dichas partes o apartados.
* Opcionalmente, Cualquier otro comentario que el autor del proyecto desee añadir.

Esta parte se puede incluir en un apartado al final del archivo de Jupyter con las soluciones propuestas.

# APÉNDICE B. COMENTARIOS ADICIONALES

* No se deberá entregar ningún programa que no esté debidamente comprobado y sea correcto.
* Cada programa estará debidamente organizado y documentado. Cada función incluida en los programas entregados deberá ser clara, estar bien organizada y estar debidamente documentada, siguiendo normas estándar. Este asunto es de gran importancia y se tendrá en cuenta fuertemente en la evaluación.
* Se sobreentiende que cada entrega está hecha exclusivamente por el autor, considerándose inaceptable entregar cualquier trabajo realizado total o parcialmente por otra persona distinta del autor firmante, o un trabajo en que se ha copiado una parte del código propuesto por otros compañeros o de cualquier otra fuente. En todo caso, se ha de consignar con total claridad la ayuda recibida o tomada prestada, por cortesía y por honradez.

# APÉNDICE C. ALGUNOS ERRORES FRECUENTES

* Organizar un programa sin usar una estructura adecuada en funciones.
* Diseñar funciones sin parámetros; diseñar funciones sin documentar; emplear identificadores inadecuados; emplear variables globales; definir funciones con constantes literales que registran valores propios del programa y se repiten en distintos lugares del mismo, tales como nombres de archivos, fechas, tamaños de muestras, etc.
* Las funciones necesarias para que nuestra aplicación trabaje no deben contener lecturas de datos (input) ni salidas (print). Los datos deben proporcionarse normalmente a través de los parámetros, y el resultado de una función, normalmente se comunica en el return, para facilitar su uso posterior. La instrucción print suele usarse más bien en instrucciones o pequeños programas de demostración. Se ha de separar el funcionamiento de nuestra aplicación de sus demostraciones de funcionamiento.
* Usar rutas absolutas: f = open('/Users/blacky/Desktop/Proyectos.txt', 'w')

Esto impide ejecutar un programa fuera del ordenador en que se ha diseñado.

Lo correcto es usar rutas relativas: f = open('./Proyectos.txt', 'r')

Tu programa debería funcionar bien cuando yo lo active en mi equipo, donde lo existe la ruta absoluta '/Users/blacky/Desktop/Proyectos.txt', por ejemplo.

* Usar archivos csv (o xlsx) para almacenar información textual plana. Cuando se genera un archivo csv, comprueba que cada fila está organizada en celdas distintas, y no todas las componentes en la primera celda, sin separador alguno.
* La presentación de un proyecto en un archivo ipynb permite ir diseñando pequeñas piezas de código, ir explicando sus porqués e ir mostrando el funcionamiento de cada una. No presentes definiciones todas de golpe, y no olvides ir mostrando el funcionamiento de cada función o cada pequeña pieza de código nueva.
* Aunque no se trate de un error, los programadores de otros lenguajes tienden a programar en Python como si fuera Java (o C++, etc.). Familiarízate con los conceptos nuevos de Python y úsalos adecuadamente: funciones que devuelven varios valores a la vez en una tupla, funciones de orden superior, listas intensionales, conjuntos, diccionarios, etc., son recursos que puedes y debes usar adecuadamente.
* Ejemplo: recorrer listas a través de su índice cuando no es necesario, sin explotar la potencia de Python.