Guía para un pequeño proyecto de programación en Python

Autor: Cristóbal Pareja Flores

Actualizado: octubre de 2024

INTRODUCCIÓN

Este pequeño resumen contiene las instrucciones de entrega de un proyecto de programación o de análisis de datos con Python y otros consejos que pueden ser útiles para realizar la entrega correctamente.

Antes de desarrollar este proyecto o cualquier otro, el primer consejo es leer el enunciado completamente, y hacerlo cuidadosamente, para evitar un esfuerzo baldío en la dirección equivocada, incluyendo este documento y el planteamiento propiamente dicho del trabajo concreto propuesto.

También quiero señalar que, en un enunciado cualquiera, se pueden producir cambios durante su desarrollo por distintos motivos: porque se descubre una situación anómala que no se previó al preparar el enunciado, o un error o ambigüedad en la descripción (así es: también yo cometo errores) o en los datos de partida o por cualquier otra causa. Esto ocurre también en cualquier proyecto real, así que, si llega el caso, conviene tomárselo con el mejor talante y aprender también de estas eventuales contingencias. Se ha de procurar informar debidamente del mejor modo posible cuando esto se produzca.

DESCRIPCIÓN DE LA ENTREGA

La entrega consistirá en una carpeta comprimida, identificada con los apellidos y nombre del estudiante (por ejemplo, "Pareja_Flores_Cristobal"), sin tildes ni eñes. Esta carpeta deberá contener los siguientes archivos:

- El archivo o los archivos con la entrega propiamente dicha (por ejemplo, "electric_cars.ipynb"), con cada uno de los apartados planteados, resueltos por ti, explicados adecuadamente, a excepción de algún apartado que requiera un programa o un módulo externo, como puede ser quizás algún programa que use la técnica map-reduce, que posiblemente se deba desarrollar aparte y consistirá en un programa en Python (ej., "principios_activos_de_medicamentos.py").

- Cada apartado suele tener un rótulo y una explicación, en modo markdown, una o varias celdas con espacio para que el estudiante aporte la solución y uno o varios ejemplares de prueba. Al completar la solución, se deberá respetar la estructura propuesta.

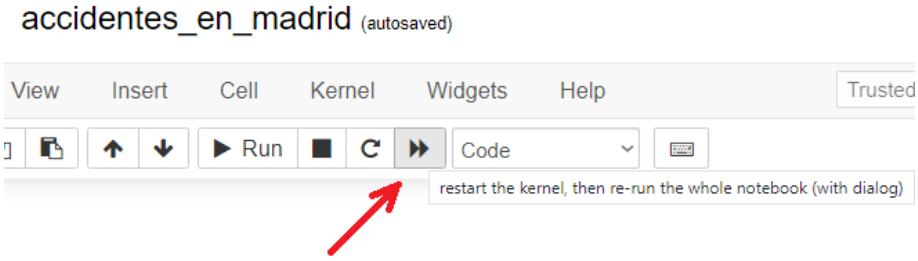
  Algún apartado se puede resolver en varios pasos con distintas funciones. Si se considera necesario, se pueden añadir celdas adicionales. Cada función deberá ir acompañada de unos pocos ejemplares de prueba bien elegidos que muestren su funcionamiento.

  Se ha de atender a las celdas proporcionadas con los ejemplares de prueba. Casi siempre, aclaran enormemente cómo debe funcionar el apartado

solicitado. Estas celdas de prueba deben respetarse escrupulosamente sin modificación alguna en ningún caso, y las funciones proporcionadas como solución por el estudiante deben satisfacer dichas pruebas de manera que el funcionamiento sea exactamente el previsto.

En cada apartado se aportará una explicación de lo logrado y de lo no logrado.

La entrega debe contener las celdas del archivo ipynb ejecutadas en orden, numeradas correlativamente, desde la primera [1] hasta la última. Esto exige una ejecución final del notebook completo antes de la entrega final:



- El archivo se entregará también en formato pdf (por ejemplo, "medicamentos.pdf"). Ambos archivos han de coincidir estrictamente. Si surge una modificación posterior del archivo ipynb o una ejecución posterior, alterándose el orden de las celdas, se deberá guardar de nuevo y, después, generar de nuevo el pdf. En el archivo ipynb (y en el pdf correspondiente) se han de poder ver las comprobaciones parciales de cada uno de los apartados.

  Debes comprobar con la antelación suficiente que puedes realizar la conversión a pdf y la preparación de la entrega según estas indicaciones.

- Para el apartado de map-reduce, debes incluir en tu carpeta el programa o los programas necesarios en Python, ejecutables desde la línea de comandos, que implementan las tareas realizadas.

  En este caso, desde el cuaderno ipynb deberá realizarse la llamada a dicho programa y, en una celda aparte, el listado del mismo:

```
! python principios_activos_de_medicamentos.py -q datos.csv
```

```
print("principios_activos_de_medicamentos.py")
```

- Los archivos de datos completos, de manera que permitan ilustrar todas las características de las soluciones aportadas. También las imágenes deben aportarse, para que la compilación por parte del profesor del archivo dé lugar a un ejemplar completo del proyecto en perfectas condiciones.

- En el mismo archivo de Jupyter de la solución, se deben incluir los apartados finales siguientes:

  - Datos personales

  - Una ficha de autoevaluación, indicando los apartados logrados, junto con la nota estimada por el estudiante en cada uno de dichos apartados.

  - La ayuda recibida o fragmentos de código tomados de otro estudiante o de cualquier otra fuente, junto con la bibliografía utilizada o páginas de Internet en que se ha encontrado algún fragmento de código inclñuiro en tu trabajo.

  - Opcionalmente, cualquier otro comentario que desees añadir.

No deberá haber más que un único cuaderno ipynb y un único pdf correspondiente a dicha solución.

COMENTARIOS ADICIONALES

- No se deberá entregar ningún programa que no esté debidamente comprobado y sea correcto. Se recomienda encarecidamente la ejecución final del notebook completo, desde el inicio (es decir, reinicnıiando el kernel), antes de la entrega final para asegurarse de que todo funciona como es debido, tal como se indicó antes.

- Cada función deberá ser clara y estar debidamente estructurada y documentada, siguiendo normas estándar. V. por ejemplo, pep-8:

  PEP 8 -- Style Guide for Python Code

  Este asunto es de gran importancia y se tendrá en cuenta fuertemente en la evaluación.

  El docstring en Python se ha de situar **debajo** de la cabecera de la función (no antes, como es costumbre en otros lenguajes). Ha de consistir en una cadena larga de caracteres (con tres comillas o dobles comillas como delimitadores).

  Los tipos de los parámetros y del resultado de las funciones puede indicarse mediante anotaciones o en el docstring, opcionalmente, pero no de ambos modos, y en todo el cuaderno del mismo modo. Aunque los tipos de datos se aporten como anotaciones, en el docstring aparecerá el cometido de la función (sin explicar cómo opera), el papel de los parámetros, sus requisitos (cuando los haya) y la posibilidad de que se dispare alguna excepción definida por ti.

  Los comentarios insertados en el código pueden ayudar a la legibilidad del mismo. Pero deben ser escuetos, y deben evitarse los comentarios obvios.

- 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. ALGUNOS ERRORES FRECUENTES

- Es erróneo…
  - Organizar un programa sin usar una estructura adecuada en funciones.
  - Diseñar funciones sin parámetros cuando pueden usarse en distintas situaciones
  - Diseñar funciones sin documentar er manera estándar
  - Emplear identificadores inadecuados. Un error típico es usar identiicadores para los parámetros que luego son los nombres de los parámetros que se van a usar.
  - Emplear variables globales. En cambio, el uso de constantes globales es adecuado.
  - 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, tablas de conversión, etc.
  - Importar librerías que no se usan.

- Las funciones necesarias para que nuestra aplicación trabaje no deben contener lecturas de datos (input), y muy pocas veces salidas mediante 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.

- Es errróneo el uso de rutas absolutas:

```
f = open('C:/Users/blacky/Desktop/Proyectos.txt', 'r')
```

  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 la carpeta de trabajo de mi equipo, donde no existe una ruta absoluta como la tuya.

- Es erróneo 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 las definiciones todas de golpe, y no olvides ir mostrando el funcionamiento de cada función o cada pequeña pieza de código nueva con las pruebas de funcionamiento necesarias.

- Los programadores procedentes 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, persiguiendo en primer lugar la claridad:

  - Recorridos de listas sin mencionar los índices
  - Recorridos de listas o diccionarios comparando la clave con la buscada, para acceder a un elemento.
  - Funciones que devuelven varios valores a la vez en una tupla
  - Asignaciones múltiples
  - Funciones de orden superior
  - Listas intensionales
  - Conjuntos, diccionarios, etc.

- Una fuente de ineficiencia frecuente es cargar un archivo completo en la memoria principal, siendo innecesario. Dos ejemplos:

  - def select_line(file_in, num_line)

    ... readlines() ...

  - def normalize_data(file_in, file_out)

  En el primer ejemplo, puede recorrerse el archivo de entrada hasta llegar a la línea que se pide, y nunca hace falta tener más de una línea en la memoria. En la función que normaliza datos, el archivo puede leerse y procesarse línea a línea, y nunca se requiere tener más de una en la memoria principal.

- Por último, pregunta las dudas que encuentres en clase o en el foro, y participa también aportando tus propios comentarios o propuestas de solución. Un error frecuente es precisamente no plantear las dudas que surjan.

C.