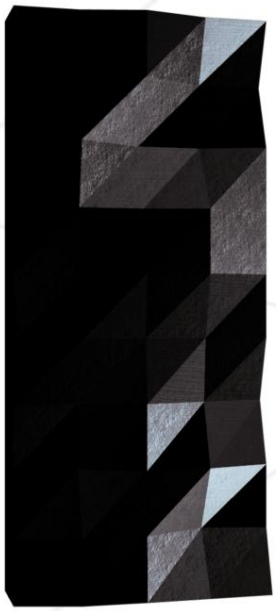

Entrenamiento de Python-Django - 1

Bibliotecas

Resumen: Hoy aprenderemos a manejar algunas bibliotecas que pueden resultar útiles en Python.

Versión: 1.1

Contenido

Capítulo I

Preámbulo

Geohashing De Wikipedia, la enciclopedia libre

Geohashing es una actividad recreativa al aire libre inspirada en el webcomic xkcd, en en el que los participantes tienen que llegar a una ubicación aleatoria (elegida por un algoritmo informático), Demuestren su logro tomando una foto de un receptor del Sistema de Posicionamiento Global (GPS) u otro dispositivo móvil y luego cuenten la historia de su viaje en línea. Basado en pruebas. En la navegación no electrónica también es aceptable.

Mientras que otras actividades recreativas al aire libre como el geocaching tienen un objetivo preciso, el geohashing se alimenta principalmente de su inutilidad, lo que resulta divertido para sus jugadores. La comunidad y cultura de geohashing resultante es, por lo tanto, extremadamente irónica, y apoya cualquier tipo de comportamiento humorístico durante la práctica del geohashing, lo que da como resultado una Parodia de las actividades tradicionales al aire libre. Navegar a un punto aleatorio no tiene por qué ser inútil. Algunos geohashers documentan las nuevas características cartográficas que encuentran en OpenStreetMap. proyecto.

Fuente Wikipedia

Capítulo II

Reglas generales

- Su proyecto debe realizarse en una máquina virtual.
- Su máquina virtual debe tener todo el software necesario para completar su proyecto. Estos software deben configurarse e instalarse.
- Puede elegir el sistema operativo que utilizará para su máquina virtual.
- Debe poder utilizar su máquina virtual desde una computadora del clúster.
- Debe utilizar una carpeta compartida entre su máquina virtual y su máquina host.
- Durante tus evaluaciones utilizarás esta carpeta para compartir con tu repositorio.

Sus funciones no deben cerrarse inesperadamente (fallo de segmentación, error de bus, doble liberación, etc.), salvo por comportamientos indefinidos. Si esto ocurre, su proyecto se considerará no funcional y recibirá un 0 en la evaluación.

Te animamos a crear programas de prueba para tu proyecto, aunque este trabajo no tenga que entregarse ni se califique. Esto te permitirá evaluar fácilmente tu trabajo y el de tus compañeros. Estas pruebas te resultarán especialmente útiles durante la defensa. De hecho, durante la defensa, puedes usar tus propias pruebas o las del compañero que estás evaluando.

Envía tu trabajo al repositorio Git asignado. Solo se calificará el trabajo en el repositorio Git. Si Deepthought se encarga de calificar tu trabajo, lo hará después de la evaluación por pares. Si se produce un error en alguna sección de tu trabajo durante la calificación de Deepthought, la evaluación se detendrá.

Capítulo III

Reglas específicas de hoy

- No hay código en el ámbito global. ¡Crea funciones!
- Cada archivo entregado debe finalizar con una llamada de función en una condición idéntica a:

```
si __nombre__ == '__principal__':
    your_function(cualquiera, parámetro, es, requerido)
```

- Puede configurar una gestión de errores en esta condición.
- Están prohibidas las importaciones, salvo las especificadas en las “Funciones Autorizadas” sección en el encabezado de cada ejercicio.
- Utilizarás el intérprete de Python3.

Capítulo IV

Ejercicio 00

	Ejercicio 00
Ejercicio 00: Antigravedad	
Directorio de entrega: ex00/	
Archivos a entregar: geohashing.py	
Funciones permitidas: módulo sys y antigravedad	

Este es un pequeño ejercicio de calentamiento que recuerda el preámbulo de hoy. Nada demasiado complicado.

Cree un pequeño programa llamado geohashing.py que tomará tantos parámetros como sean necesarios para calcular un geohash típico y, obviamente, debería calcular este geohash antes de mostrarlo en la salida estándar.

En caso de error, el programa deberá mostrar un mensaje relevante que usted habrá elegido antes de salir correctamente.

Este esquema podría ayudar: Algoritmo Geohashing.

Capítulo V

Ejercicio 01

	Ejercicio 01
Ejercicio 01: Pip	
Directorio de entrega: ex01/	
Archivos a entregar: my_script.sh my_program.py Funciones	
permitidas: ruta del módulo (anteriormente llamado path.py)	

path es una biblioteca que implementa un objeto Path alrededor del módulo os.path de Python, haciendo su uso muy intuitivo.

En este ejercicio, creará un script bash que instala esta biblioteca, así como un Programa Python que lo utiliza.

El script de Shell debe ajustarse a esta descripción:

- Su nombre debe tener una extensión .sh porque es un script de Shell.
- Debe mostrar qué versión de pip utiliza.

Debe instalar la versión de desarrollo de path.py desde su repositorio de GitHub, en una carpeta llamada local_lib, ubicada en la carpeta del repositorio. Si la biblioteca ya está instalada en la carpeta, la instalación debe eliminarla.

- Debe escribir los registros de instalación de la ruta en un archivo con extensión .log.
- Si la biblioteca se ha instalado correctamente, debe ejecutar el pequeño programa que han creado.

El programa Python que debes crear es una composición de tu elección que debe, como sigue:
Observe siempre estas restricciones:

- Su extensión debe ser .py porque es un Python.
- Se debe importar el módulo path desde la ubicación donde se haya instalado esta librería gracias al script anterior.
- Debe crear una carpeta y un archivo dentro de esta carpeta, escribir algo en este archivo y Luego lea y muestre su contenido.
- Deberá respetar las normas específicas del día.

Capítulo VI

Ejercicio 02

	Ejercicio 02
Ejercicio 02: solicitar una API	
Directorio de entrega: ex02/	
Archivos a entregar: request_wikipedia.py requirements.txt Funciones permitidas:	
módulos requests, json, dewiki y sys	

Wikipedia es una increíble herramienta compartida que seguramente ya conoces. Está disponible en tu navegador favorito y como aplicación móvil. Te invitamos a crear una herramienta que te permita acceder a este sitio web esencial, ahora directamente desde tu dispositivo.

Para ello, debes diseñar un programa llamado request_wikipedia.py que tome una cadena como parámetro y realice una búsqueda a través de la API de Wikipedia. Antes de escribir el resultado en el archivo, puedes solicitar la API en francés o inglés.

El programa debe escribir un resultado, incluso si la solicitud está mal escrita. Tomemos como ejemplo el sitio web original: si encuentra un resultado para una solicitud dada, su programa también debería hacerlo.

- El resultado no debe estar formateado en JSON o Wiki Markup antes de escribirse en el archivo.
- El nombre del archivo debe tener el siguiente formato: nombre_de_la_búsqueda.wiki y no debe contener ningún espacio.
- En caso de ausencia de parámetros, parámetro erróneo, solicitud no válida, información no encontrada, problema con el servidor o cualquier otro problema: no se debe crear ningún archivo y se debe mostrar un mensaje de error relevante en la consola.

Incluya el archivo requirements.txt en su repositorio. Se utilizará durante la evaluación para instalar las bibliotecas necesarias en su programa VirtualEnv o en el sistema.

La biblioteca dewiki no es perfecta. No buscamos el mejor resultado; ese no es el objetivo de este ejercicio.

Lea atentamente la documentación de la API. Observe la estructura que se le envía.

He aquí un ejemplo de lo que se espera:

```
$>python3 request_wikipedia.py "chocolatina" $>gato chocolatine.wiki



Une chocolatine designe: * une
viennoiserie au chocolat, aussi appelee Pain au chocolat ou couque au chocolat; * une viennoiserie a la crème patissiere et au
chocolat, aussi appelee suisse; * una especie de bombón de chocolate; * un ultraje de Anna Rozen



Malgre son use ancien, le mot n'est entre dans le dictionnaire Petit Robert qu'en 2007 et dans le Petit Larousse qu'en 2011.


El uso del término "Chocolatina" se retrouve igualmente en Quebec, no evolucione a partir del vieux francais diferenciándose de los franceses empleados en
        Europa, pero este uso no demuestra ni n'infirme aucune anteriorite, depende du hasard de l'usage du premier commercant l'ayant introduit au
      Quebec.


 Referencias



Categoría:Pastelería
Categoría:Chocolate
```

Capítulo VII

ejercicio 03

	Ejercicio 03
Ejercicio 03: Analizador HTML	
Directorio de entrega: ex03/	
Archivos a entregar: roads_to_philosophy.py, requirements.txt Funciones permitidas:	
módulos sys, requests y BeautifulSoup	

Cuenta la leyenda que si partes de cualquier artículo de Wikipedia, haces clic en el primer enlace de la introducción de ese artículo (que no está en cursiva ni entre corchetes) y repites el proceso improvisando, acabarás siempre en el artículo sobre Filosofía.

Bueno, ¡esto no es una leyenda! (Por favor, quédense atónitos). Como lo demuestra este ... artículo de Wikipedia.

Pero como sólo crees en lo que ves con tus propios ojos, debes crear un programa que pruebe este fenómeno enumerando y contando todos los artículos entre tu solicitud y el artículo de Wikipedia: los caminos de la filosofía.

Este programa debe llamarse roads_to_philosophy.py y comportarse de la siguiente manera:

- El programa debe tomar como parámetro una cadena que sea una palabra o un grupo de palabras que coincidan solo con una búsqueda en Wikipedia.
- El programa debe solicitar una URL de Wikipedia en inglés idéntica a una estándar Buscar en un navegador. En otras palabras: no puedes usar la API del sitio.

- Debe analizar la página html gracias a la biblioteca BeautifulSoup para:
  - Encontrar la redirección eventual y tomarla en cuenta en los caminos hacia la phi-Losofía. Ojo, no es una redirección de URL.
  - Busca el título principal de la página y agrégalo a los caminos de la filosofía.
  - Buscar (si existe) el primer enlace del párrafo de introducción que lleva a otro artículo de Wikipedia. En lugar de ignorar lo que está en cursiva o entre corchetes, el programa debe ignorar cuidadosamente los enlaces que no dirigen a un nuevo artículo, como los que llevan a la sección de ayuda de Wikipedia.
- El programa debe comenzar nuevamente desde el paso 2 a partir del enlace obtenido durante el paso anterior hasta llegar a una de esas ocurrencias:

  El enlace lleva a la página de filosofía. De ser así, debe imprimir los artículos visitados, así como el número total de estos artículos, en el siguiente formato: <número> caminos desde <solicitud> a filosofía en la salida estándar.
  - La página no incluía ningún enlace válido. El programa debe mostrar: Conduce ¡A un callejón sin salida!.

  El enlace lleva a una página ya visitada, lo que significa que el programa está a punto de repetirse indefinidamente. De ser así, el mensaje mostrado debe ser: ¡Conduce a un bucle infinito!

- En esta etapa, después de mostrar los mensajes necesarios en la salida estándar, el El programa debe cerrarse correctamente.

Si en cualquier momento durante la ejecución del programa se produce un error como un error de conexión, de servidor, de parámetro, de solicitud o cualquier otro tipo de error, el programa debe cerrarse correctamente con un error relevante.
mensaje.

Al igual que en el ejercicio anterior, debes proporcionar un archivo requirements.txt con tu solicitud. grama para facilitar la instalación de bibliotecas.

La salida de su programa debe verse así:

```
$> python3 roads_to_philosophy.py "42 (número)" 42 (número)

Número natural
Matemáticas
Griego antiguo
idioma griego
Griego moderno
Palabra familiar
Palabra
Lingüística
Ciencia
Conocimiento
Conciencia
Consciente
Conciencia
Calidad (filosofía)
Filosofía: ¡17
caminos del 42 (número) a la filosofía! $> python3 roads_to_philosophy.py
Accuvio ¡Es un callejón sin salida! $>
```

La comunidad de Wikipedia actualiza los artículos con frecuencia. Es muy probable que, entre la creación de este tema y el día en que lo tomes, los caminos hacia la filosofía hayan cambiado y el ejemplo de Accuvio ya no sea un callejón sin salida.

Capítulo VIII

Ejercicio 04

	Ejercicio 04
Ejercicio 04: Virtualenv	
Directorio de entrega: ex04/	
Archivos a entregar: requirements.txt my_script.sh Funciones	
permitidas: everything	

Mañana comenzarás tu formación en el framework Django. Debes sentar las bases.
configurando una pequeña instalación sencilla.

Para ello, crearás dos elementos:

- Un archivo requirements.txt que debe incluir las últimas versiones estables de django y psycopg2.
- Un script con el siguiente comportamiento:
  - Tener la extensión .sh.
  - Cree un entorno virtual en Python3 llamado django_venv.
  - Instale el archivo requirements.txt que ha creado en VirtualEnv.
  - El entorno virtual debe estar activado al salir.

Capítulo IX

Ejercicio 05

	Ejercicio 05
Ejercicio 05: Hola mundo	
Directorio de entregas: ex05/	
Archivos a entregar: cualquier archivo necesario	
Funciones permitidas: todas	

Debe ser frustrante simplemente instalar Django. Te entendemos.

Por eso, este día sobre bibliotecas terminarás con un éxito rotundo, diseñando tu primera biblioteca. Hola mundo con Django.

En este ejercicio final, deberás seguir y adaptar el tutorial oficial para crear una página web que simplemente muestre el texto ¡Hola mundo! en el navegador en la siguiente dirección: http://localhost:8000/helloworld.

Su repositorio debe ser una carpeta que contenga el proyecto Django.

Capítulo X

Presentación y evaluación por pares

Entrega tu tarea en tu repositorio de Git como de costumbre. Solo se evaluará el trabajo dentro de tus repositorios durante la defensa. No dudes en verificar los nombres de tus carpetas y archivos para asegurarte de que sean correctos.

El proceso de evaluación se realizará en el computador del evaluado.
grupo.