Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2018-2

22 de noviembre de 2018
Evaluada

Actividad 14

Web Services & Regex

Introducción

Dado el **reciente descubrimiento de agua en Marte** la preocupación de los marcianos por posibles *tsunamis* se elevó a cifras históricas. Esto causó que abdujeran al malvado *Dr. Herny*, al ser confundido con el famoso geógrafo, *Marcelo Lakes*, mientras paseaba junto a su secuaz *"El Tini"*. Para tratar de encontrar el paradero del *Dr. Herny*, *"El Tini"* pidió ayuda a la NASA, y le dieron acceso a sus registros fotográficos. Por ahora, la única pista que tienes, es un archivo que contiene fechas en las que el *Dr. Herny* pudo haber sido fotografiado, pero este fue corrompido por *hackers* marcianos. Tu labor será limpiar el ataque de los extraterrestres y ocupar la información para descargar estas imágenes y buscar al malvado *Dr. Herny*.

1. Instrucciones generales

En esta actividad deberás limpiar la información que se encuentra en el archivo `fechas_secretas.txt` y extraer las fechas válidas que están allí. Luego, estas serán utilizadas para consultar la API de APOD (*Astronomical Picture of the Day*) de la NASA, para obtener las imágenes correspondientes a dichas fechas.

2. *Regex*

Para poder ver las imágenes, primero deberás extraer las fechas en formato correcto (`aaaa-mm-dd`) del archivo `fechas_secretas.txt`. Cada una de las líneas de este archivo contiene una fecha, la cual fue corrompida y escondida entre etiquetas (o *tags*) de HTML.

Deberás rellenar las siguientes funciones que se encuentran en el archivo `main.py` para obtener las fechas a utilizar en el siguiente paso:

- `def limpiar_fecha(linea)`: Esta función recibe un *string*, el cual corresponderá a una línea del archivo `secret_dates.txt`. Deberás limpiarla de las etiquetas que correspondan según se describe a continuación y retornar el *string* resultante.

  Debes filtrar **solamente** las etiquetas de HTML válidas. Para esto debes tener en cuenta que una etiqueta válida cumple con el siguiente formato:

  - Cada etiqueta comienza con el carácter `<` y termina con `>`.

- Las etiquetas son distintas dependiendo si son etiquetas de inicio o de cierre. Una etiqueta de inicio se ve de la siguiente manera: `<h1>`, mientras que su correspondiente etiqueta de cierre sería: `</h1>`.
- El tipo de la etiqueta (por ejemplo, `h1` de una etiqueta `<h1>`) siempre debe estar compuesto de caracteres alfanuméricos. Claro, únicamente letras y números pertenecientes a ASCII.

Nota: Cabe notar que, luego de una etiqueta de inicio, no necesariamente existirá una etiqueta de cierre. Por esto, sólo debes preocuparte individualmente de cada una de ellas. Además, no te encontrarás con etiquetas que posean atributos y debes ignorar este caso.

Debes considerar que te puedes encontrar con etiquetas que no son válidas y que **no deben** ser eliminados. Una etiqueta inválida es aquella que no cumple con una o más de las condiciones anteriores. Por ejemplo, `<//span>`, `<br?>` y `<b o d y>` no son etiquetas válidas.

Un ejemplo de como se vería una línea del archivo y luego de ser filtrada:

```
<h3>20<br><?p!>09-03-<br>05<?/h3!>
```

```
20<?p!>09-03-05<?/h3!>
```

- `def chequear_fecha(fecha)`: Dado que no todas las fechas tienen un formato válido. Aquí debes chequear que la fecha esté **limpia** y tenga en el **formato correcto**. Esta función recibe un *string* con un candidato a fecha y retorna un booleano que será `True` si en caso de que el candidato cumpla con un formato válido de fecha.

  El formato de la fecha corresponde a `aaaa-mm-dd`, donde `a`, `m` y `d` corresponden a dígitos decimales.[1] Un ejemplo de fecha con formato correcto es

  ```
  2018-11-22
  ```

  Mientras que los siguientes corresponden a fechas con formato incorrectos,

  ```
  20aa-03-05
  ```

  ```
  20<?p!>09-03-05<?/h3!>
  ```

- `def obtener_fechas(path)`: Recibe el *path* hacia el archivo con las fechas ocultas. Debes ocupar las dos funciones hechas con anterioridad para limpiar el archivo y obtener las fechas. Esta función debe retornar un iterable con las fechas válidas, que podrán ser ocupadas para las consultas a la API.

¡Importante! Para realizar el procesamiento de los *strings* en estas dos funciones, **debes** ocupar los métodos de la librería `re`. Está **prohibido** utilizar métodos de la clase `str`.

3. *Web services*

Ahora para descargar las imágenes, debes utilizar la API de APOD (*Astronomical Picture of the Day*) de la NASA (la documentación la pueden encontrar **aquí**). Esta te permitirá obtener el URL e información extra de las fotos del día que publicó la NASA en las fechas que obtuviste anteriormente.

Para esto, deberás trabajar con las siguientes funciones que están en el archivo `main.py`:

[1] Una vez que limpies las fechas y verifiques su formato, puedes asumir que **estas son correctas**. Por ejemplo, **no** te entregaremos el 31 de febrero del 2050 o 2010-50-50.

- `def obtener_info(fecha)`: Esta función recibe una fecha. En base a esta, debe ejecutar una consulta a la API —tal y como se describe a continuación— y retornar un diccionario que contenga el título, la fecha y el URL de la imagen del día de la fecha consultada.

  Para obtener la información debes generar una solicitud (o *request*) utilizando el método `GET` de `HTML` dirigida a `https://api.nasa.gov/planetary/apod`, con los siguientes parámetros,

Parámetro	Descripción	Formato
`date`	Fecha que quieres consultar	`str: aaaa-mm-dd`
`hd`	Determina la calidad de la imagen	`bool`: por defecto, `False`
`api_key`	Es la autenticación que requiere la API	`str`

  En cada solicitud debes consultar por las imágenes en baja resolución en las fecha que se te indica. Deberás ocupara la API *key* que obtuviste en tu registro (si no lo hiciste, debes registrarte aquí).

  ¡Importante! Debes crear un archivo `credenciales.py` donde almacenarás tu *key* que luego debe estar importado en `main.py`. Este archivo **no** lo debes agregar a tu repositorio, ya que otras personas que tengan acceso a tu repositorio podrían acceder a tu API *key* y utilizarla de manera no apropiada.

  Cada solicitud que hagas a la API de la forma descrita, te entregará una respuesta en formato **JSON** de siguiente forma,

```
{
    "copyright": "Juan Carlos Casado",
    "date": "2010-12-12",
    "explanation": "In 1999, Leonids Meteor...",
    "hdurl": "https://apod.nasa.gov/apod/image/1012/leonids99_casado_big.jpg",
    "media_type": "image",
    "service_version": "v1",
    "title": "Leonids Above Torre de la Guaita",
    "url": "https://apod.nasa.gov/apod/image/1012/leonids99_casado.jpg"
}
```

  Debes retornar los datos pedidos en base a esta respuesta.

- `def escribir_respuesta(datos)`: Esta función recibe un diccionario con información del título, fecha y URL de una imagen y guarda esta información en el archivo `resultados.txt`. En este archivo, cada línea debe tener la información de una imagen y debe ser de la forma: `"{fecha} --> {titulo}: {url}"`. Además esta función guarda cada una de las imágenes en tu computador[2], dentro de la carpeta `imagenes` de tu repositorio. La imagen debe tener su nombre original; es decir, para el siguiente URL: `https://apod.nasa.gov/apod/image/1012/leonids99_casado.jpg`, el nombre de la imagen debe ser `leonids99_casado.jpg`.

Uso de `.gitignore`

Para esta actividad debes utilizar un archivo `.gitignore` para no subir el archivo `credenciales.py` y las imágenes descargadas[3] a tu repositorio remoto. De lo contrario, tu actividad sufrirá un descuento de **tres**

[2] Te entregamos la función `def descargar_imagen` que te podría ser de ayuda.
[3] Ten en cuenta que las extensiones más comunes de imágenes son `.jpg`, `.png` y `.gif`

décimas por el uso incorrecto de `.gitignore` y **tres décimas** más por subir tu credencial e imágenes al repositorio. **Este enlace** te puede ayudar sobre "cómo eliminar del repositorio remoto" un archivo, luego de agregarlo al `.gitignore`.

Requerimientos

- (2,5 pts) Uso de *regex*
  - (1,0 pt) Implementación `limpiar_fecha`
  - (1,0 pt) Implementación `chequear_fecha`
  - (0,5 pts) Implementación `obtener_fechas`
- (3,5 pts) *Web services*
  - (2,5 pts) Implementación `obtener_info`
    - (0,5 pts) Uso correcto de `credenciales.py`
    - (0,5 pts) Realizar la *request* con el método y dirección correctos
    - (0,5 pts) Enviar los argumentos correctos a la API
    - (1,0 pt) Obtener correctamente la información desde la respuesta de la API
  - (1,0 pt) Implementación `escribir_respuesta`
    - (0,3 pts) Guardar resultados en `resultados.txt`
    - (0,2 pts) Escribir los resultados con el formato especificado
    - (0,5 pts) Descarga correcta de las imágenes

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC14/`
- **Hora del último *push* válido:** 16:20