Laboratorio 1: Uso de Expresiones Regulares en Python

Teoría de Lenguajes

3 de abril de 2024

1 Introducción

El objetivo de este laboratorio es el uso de expresiones regulares para dar soluciones a problemas que se podrían encontrar en la práctica al trabajar con archivos o entradas de texto. Las expresiones regulares pasarán a ser una herramienta a tener en cuenta para utilizar en el tratamiento de textos, como archivos de texto en lenguaje natural, logs de programas, y archivos de código, entre muchos otros. Podrá utilizarlas de forma sencilla en lenguajes de programación (ej. pearl, python, ruby, etc.), a través de comandos de terminal (ej. grep, sed, awk, etc.) o editores de texto.

En este laboratorio se pide escribir un conjunto de rutinas en el lenguaje de programación Python 3. Se espera que los problemas sean resueltos con expresiones regulares, evitando utilizar estructuras de control (ej. for, while, if, etc.) u otras herramientas para resolverlos.

2 Modo de trabajo

- Se indican un conjunto de programas a realizar
- Se imparte para cada programa **archivos de entrada** y **archivos de salida** que contienen la salida esperada para cada entrada.

- Cada grupo deberá implementar los programa pedidos considerando la especificación en esta letra, las entradas y sus correspondientes salidas.
- La salida del programa del grupo deberá ser **exactamente igual** a la salida de referencia. También se deberán mantener los nombres de los archivos de entrada, salida, y de los programas.
- Cada grupo debe entregar los programas implementados y el archivo "integrantes.txt" que se detalla más adelante en esta letra.

3 Realidad planteada

Una aplicación popular en el área de Procesamiento de Lenguaje Natural es el análisis y moderación de mensajes como los que puede haber en un blog; particularmente en secciones de foros donde los usuarios pueden iniciar conversaciones y participar en discusiones. Estos foros de blogs presentan una rica variedad de interacciones humanas, desde preguntas y respuestas hasta debates extensos sobre temas específicos.

En este laboratorio, nos enfocaremos en el uso de expresiones regulares — una herramienta poderosa y versátil para el procesamiento de texto — para abordar desafíos comunes en la gestión de estos foros. Las expresiones regulares nos permitirán realizar tareas como la búsqueda y filtrado de mensajes mediante la identificación de patrones.

4 Sintaxis de archivos de entrada

Cada archivo JSON [5] contiene una lista de mensajes. Para cada mensaje se tiene los siguientes campos: `"user"`, que indica el usuario que escribió el mensaje, `"timestamp"`, que indica cuándo fue enviado el mensaje y `"content"` que contiene el mensaje que se envió.

El contenido del mensaje puede contener algunas expresiones del lenguaje Markdown [6]:

- **texto en negrita**
- *texto en cursiva*

- #encabezado1
- ##encabezado2
- ###encabezado3
- ~~texto tachado~~

Además el mensaje puede contener:

- menciones a otros usuarios (indicados por @usuario_a_mencionar)
- fines de línea (indicados por \n)

4.1 Ejemplo de archivo (0.json)

```
{
  "messages": [
    {
      "user": "usuario1",
      "timestamp": "T 2023:03:15 08:00:00",
      "content": "#Python\n Hola, estoy aprendiendo
    expresiones regulares en python y me gustaria que me
    recomienden como empezar"
    },
    {
      "user": "usuario2",
      "timestamp": "T 2023:03:15 10:30:00",
      "content": "Hola @usuario1!\n Para empezar lo mas **
    importante** es conocer la libreria *re*, que contiene
    todas las funciones de expresiones regulares que vas a
    precisar!"
    },
    {
      "user": "usuario3",
      "timestamp": "T 2023:03:15 12:45:00",
      "content": "Totalmente de acuerdo con @usuario2, ademas
    , considero importante que utilices python3 y no ~~python2
    ~~"
    },
    {
      "user": "usuario4",
      "timestamp": "T 2023:03:15 14:30:00",
```

```
21        "content": "Interesante conversacion @usuario1
      @usuario2 @usuario3.\n Agregaria que el uso de **
      expresiones regulares** es muy comun hoy en dia y que vas
      a encontrar muchas guias en distintos foros online."
22      },
23      {
24        "user": "usuario1",
25        "timestamp": "T 2023:03:16 09:15:00",
26        "content": "##Muchas gracias\n Gracias a todos por sus
      consejos! Python ~~no~~ es genial"
27      }
28    ]
29  }
```

En cada programa se indicará una operación a realizar con el archivo de entrada, con un formato como el anterior, y la salida que debe desplegar. Recordamos que **la solución debe priorizar el uso de expresiones regulares** por sobre otras estructuras de control.

5 Programas a implementar

Implemente en Python 3 utilizando el módulo de expresiones regulares ***re*** los siguientes programas:

5.1 programa0.py

Despliega el nombre del usuario que inició la conversación.

Este programa será dado como ejemplo.

Con el archivo de ejemplo (0.json) se obtiene la siguiente salida:

```
usuario1
```

5.2 programa1.py

Despliega el nombre de todos los usuarios que son mencionados en algún mensaje. Es decir que se utiliza el caracter @ para mencionarlos en alguno de los mensajes de la entrada.

Con el archivo de ejemplo (0.json) se obtiene la siguiente salida:

```
usuario1
usuario2
usuario3
```

5.3 programa2.py

Despliega el nombre de todos los usuarios que escribieron en la conversación y cuántas veces lo hicieron.

Con el archivo de ejemplo (0.json) se obtiene la siguiente salida:

```
usuario1 2
usuario2 1
usuario3 1
usuario4 1
```

5.4 programa3.py

Despliega el timestamp de todos los mensajes de la conversación de la forma dd + "de" + mm + "del" + aaaa + " a las " + hh:mm + " hs.".

Con el archivo de ejemplo (0.json) se obtiene la siguiente salida:

```
15 de marzo del 2023 a las 08:00 hs
15 de marzo del 2023 a las 10:30 hs
15 de marzo del 2023 a las 12:45 hs
15 de marzo del 2023 a las 14:30 hs
16 de marzo del 2023 a las 09:15 hs
```

5.5 programa4.py

Despliega los encabezados (de cualquier tipo) de los mensajes, en orden de aparición en el json (se devuelve solo el texto, sin los #).

Con el archivo de ejemplo (0.json) se obtiene la siguiente salida:

Python
Muchas gracias

5.6 programa5.py

Despliega el json traduciendo los mensajes del lenguaje Markdown a formato HTML, cambiando:

- #texto por <h1>texto</h1> (notar que el # termina cuando se encuentra un \n)
- ##texto por <h2>texto</h2> (notar que el ## termina cuando se encuentra un \n)
- ###texto por <h3>texto</h3> (notar que el ### termina cuando se encuentra un \n)
- **texto** por texto
- *texto* por texto
- ~~texto~~ por <s>texto</s>

Con el archivo de ejemplo (0.json) se obtiene la siguiente salida:

```
{
  "messages": [
    {
      "user": "usuario1",
      "timestamp": "T 2023:03:15 08:00:00",
      "content": "<h1>Python</h1>\n Hola, estoy
    aprendiendo expresiones regulares en python y me
    gustaria que me recomienden como empezar"
    },
    {
      "user": "usuario2",
      "timestamp": "T 2023:03:15 10:30:00",
      "content": "Hola @usuario1!\n Para empezar lo mas <
    strong>importante</strong> es conocer la libreria <em>
    re</em>, que contiene todas las funciones de
    expresiones regulares que vas a precisar!"
```

```
    },
    {
      "user": "usuario3",
      "timestamp": "T 2023:03:15 12:45:00",
      "content": "Totalmente de acuerdo con @usuario2,
    ademas, considero importante que utilices python3 y no
    <s>python2</s>"
    },
    {
      "user": "usuario4",
      "timestamp": "T 2023:03:15 14:30:00",
      "content": "Interesante conversacion @usuario1
    @usuario2 @usuario3.\n Agregaria que el uso de <strong>
    expresiones regulares</strong> es muy comun hoy en dia
    y que vas a encontrar muchas guias en distintos foros
    online."
    },
    {
      "user": "usuario1",
      "timestamp": "T 2023:03:16 09:15:00",
      "content": "<h2>Muchas gracias</h2>\n Gracias a
    todos por sus consejos! Python <s>no</s> es genial"
    }
  ]
}
```

6 Herramientas a utilizar

Las rutinas deben estar codificadas en Python 3 [2] (Python 3.9 [1] o superior).

Para el módulo ***re*** de expresiones regulares de Python sugerimos consultar los sitios [3] y [4].

6.1 lab1.zip

Junto con esta letra se imparte el archivo lab1.zip con el siguiente contenido:

- Los **archivos de entrada** (entradas/*.json)

- Los **archivos con las salidas esperadas de cada programa para cada entrada** (salidas_esperadas/*.txt).
- El directorio **"programas"** con el **código** de **programa0.py**. El resto de los programas deben ser incluidos en este directorio
- Un **script test.py** para ejecutar los programas y comparar las salidas.
- El programa **diff.exe para Windows** para comparar archivos.

7 Grupos

Los trabajos se deben realizar en grupos de 2 a 4 estudiantes. **No se permitirán grupos de un solo estudiante.**

Crearemos un foro en EVA para buscar integrantes faltantes para la conformación de grupos

8 Entrega

La fecha límite para la entrega es el **25 de abril a las 23:59**.

Se habilitará un formulario en EVA para realizar la entrega.

Se debe entregar:

- Los archivos **programa1,2,3,4,5.py** con los programas implementados
- Un archivo **integrantes.txt** con las cédulas (sin puntos ni dígito de verificación) y nombre de los integrantes del grupo, uno por línea, separado por comas como se muestra en el ejemplo. La primera línea debe contener la cantidad de integrantes del grupo. Opcionalmente, se puede utilizar el final de este archivo para comentarios que el grupo considere pertinentes.

Ejemplo de archivo integrantes.txt:

```
3
7123456, ApellidoA1 ApellidoA2, NombreA1 NombreA2
8654321, ApellidoB1 ApellidoB2, NombreA1 NombreB2
9876543, ApellidoC1 ApellidoC2, NombreC1 NombreC2
Hicimos dos soluciones para el programa3, entregamos la que consideramos mejor y dejamos comentada la otra.
```

9 Corrección

La corrección se realizará en el **sistema operativo Linux**. Recomendamos ejecutar los casos de prueba en Linux antes de entregar.

Si la ejecución termina abruptamente, o hay diferencia en los archivos de salida con la salida oficial, la solución no será considerada correcta.

Se deben respetar los nombres y forma de uso de los programas y el formato del archivo *integrantes.txt*. No se debe modificar el script test.py

10 Referencias

1. Python 3.9.2
2. Python 3.12.2 documentation
3. re — Operaciones con expresiones regulares
4. Python RegEx
5. JSON
6. Markdown