



INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO (ESCOM)



TEORIA COMPUTACIONAL

NOMBRE Y NÚMERO DE LA PRÁCTICA:

- PRÁCTICA 3. EXPRESIONES REGULARES

NOMBRE DEL ALUMNO:

- SANTOS MÉNDEZ ULISES JESÚS

NOMBRE DEL MAESTRO:

- JORGE LUIS ROSAS TRIGUEROS

FECHA DE REALIZACIÓN:

- 6/11/2020

FECHA DE ENTREGA:

- 13/11/2020

Marco Teórico

Lenguajes y expresiones regulares

Las expresiones regulares pueden definir de forma exacta los mismos lenguajes que describen los distintos tipos de autómatas.

Las expresiones regulares ofrecen una forma declarativa para expresar las cadenas que deseamos aceptar.

Una expresión regular (o RE, por sus siglas en inglés) especifica un conjunto de cadenas que coinciden con ella; las funciones de este módulo permiten comprobar si una determinada cadena coincide con una expresión regular dada (o si una expresión regular dada coincide con una determinada cadena, que se reduce a lo mismo).

Las expresiones regulares sirven como lenguaje de entrada de muchos sistemas que procesan cadenas, tienen algunos usos como:

Comandos de búsqueda tales como el comando grep de UNIX o comandos equivalentes para localizar cadenas en los exploradores web o en los sistemas de formateo de texto. Estos sistemas emplean una notación de tipo expresión regular para describir los patrones que el usuario desea localizar en un archivo. Los distintos sistemas de búsqueda convierten la expresión regular bien en un AFD (Autómata Finito Determinista) y simulan dicho autómata sobre el archivo en que se va a realizar la búsqueda (Véase Tabla 1).¹

[A-Za-z0-9_]	Caracteres alfanuméricos y “_”
[A-Za-z]	Caracteres alfabéticos
[\t]	Espacio y tabulador
[\t\r\n\v\f]	Espacios
[0-9]	Dígitos
[a-z]	Letras minúsculas
[A-Z]	Letras mayúsculas
[!\"#\$%&'()*+,-./:;<=>?@\\^_`{ }~--]	Caracteres de puntuación

(Tabla 1)

- “^”, inicio de línea
- “\$”, fin de línea
- “<”, principio de palabra
- “>”, fin de palabra
- “\b”, límite de palabra

En Python el módulo de terceros regex, cuenta con una API compatible con el módulo de la biblioteca estándar re, el cual ofrece una funcionalidad adicional y un soporte Unicode más completo.

¹ Tabla de expresiones regulares en Linux.

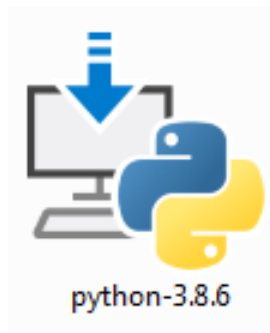
Material y Equipo:

-PC (véase figura 1)²



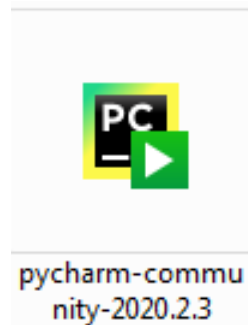
(Figura 1)

-Python 3 (Python 3.8.6) (véase figura. 2)³



(Figura 2)

-IDE (Pycharm) (véase figura 3)⁴



(Figura 3)

² Figura 1: PC o cualquier computadora para el desarrollo de la práctica.

³ Ejecutable para instalar Python 3.8.6.

⁴ Ejecutable de IDE Pycharm .

Desarrollo:

- 1) Se comenzó por ingresar al link que se encuentra en la asignación de la práctica (Véase Figura 4)⁵.

<https://regex.sketchengine.co.uk/>

(Figura 4)

- 2) Se realizó un ejercicio de prueba con el maestro para saber que tipo de caracteres se ingresan y se iluminan el conjunto o la palabra que corresponda a la expresión regular (Véase figura 5)⁶.

Exercise 1

Enter a regexp that matches all the items in the first column (positive examples) but none of those in the second (negative examples). When you press "submit", you will see what matched.

Regexp:

Positive Negative

pit	pt
spot	Pot
spate	peat
slap two	part
respite	

(Figura 5)

- 3) Se nos dejó como primer punto de la práctica resolver los 4 ejercicios que se encuentran en la página (Véase Figura 6)⁷.

Regexp:

Exercise 1

Enter a regexp that matches all the items in the first column (positive examples) but none of those in the second (negative examples). When you press "submit", you will see what matched.

Regexp:

Positive Negative

pit	pt
spot	Pot
spate	peat
slap two	part
respite	

(Figura 6)

⁵ Imagen del link de consulta de los ejercicios para la primera parte de la práctica.

⁶ Ejemplo de que tipo de comandos se ingresan para el registro .

⁷ Solución del ejercicio no. 1.

4) Segundo Ejercicio de la primera parte de la práctica (Véase Figura 7)⁸.

Regexp:

(Figura 7)

Exercise 2

Enter a regexp that matches all the items in the first column (positive examples) but none of those in the second (negative examples). When you press "submit", you will see what matched.

Regexp:

Positive	Negative
	aleht
rap them	happy them
tapeth	tarpth
apth	Apt
wrap'try	peth
sap tray	tarreth
87ap9th	ddapdg
apothecary	apples
	shape the

5) Tercer ejercicio de la primera parte de la práctica (Véase Figura 8)⁹.

Regexp:

Exercise 3

Enter a regexp that matches all the items in the first column (positive examples) but none of those in the second (negative examples). When you press "submit", you will see what matched.

Regexp:

Positive	Negative
afgfkng	fgok
rafgkahe	a fgk
bafghk	affgm
baffgkit	affhkh
afgfkng	fgok
rafgkahe	afg K
bafghk	aff gm
baffg kit	affhghk

(Figura 8)

⁸ Solución del ejercicio no. 2.

⁹ Solución del ejercicio no. 3.

6) Cuarto ejercicio de la primera parte de la práctica (Véase Figura 9)¹⁰.

Regex:

Exercise 4: Finding sentence breaks

Finding where one sentence ends and another begins is trickier than might be imagined. Enter a regex that matches all the items in the first column (positive examples) but none of those in the second (negative examples). When you press "submit", you will see what matched.

Regex:

Positive	Negative
assumes word senses. Within	in the U.S.A., people often
does the clustering. In the	John?", he often thought, but
but when? It was hard to tell	weighed 17.5 grams
he arrive." After she had	well ... they'd better not
mess! He did not let it	A.I. has long been a very
it wasn't hers! She replied	like that", he thought
always thought so.) Then	but W. G. Grace never had much

(Figura 9)

- 7) Se hace la parte dos de la práctica, consiste en utilizar el comando de búsqueda en Linux para determinar una expresión regular para identificar todas las líneas en las que no aparezca ninguna palabra con dos vocales juntas en el Himno del IPN:
- 8) Inicialmente se guarda un archivo con extensión .txt (Véase Figura 10)¹¹ para que se pueda cargar a la consola que proporciona la página <https://bellard.org/jslinux/vm.html?url=alpine-x86.cfg&mem=192>

```
himno_ipn: Bloc de notas
Archivo Edición Formato Ver Ayuda
politecnico, fragua encendida
con la chispa del genio creador
en ti forja su nueva estructura
nuestra noble y pujante nacion.
En la aurora de un dia venturoso
te dio vida la Revolucion;
una estrella te puso en las manos,
ique no apague su limpio fulgor!
(CORO)
Su libertad
Mexico crea,
surge la Patria
nace la luz;
nos convoca tu voz, Politecnico,
nos conduce tu amor, juventud.
En dinamico anhelo conjugas
las dos fuerzas de un mundo viril:
es la ciencia crisol de esperanzas
es la tecnica impulso motriz.
Guinda y blanco, indomita almena
que defiende tu ardor juvenil,
oriflama en las lides gallardas
en tus manos triunfal Banderin.
(CORO)
Tus brigadas de nitida albura
ciencia augusta, saber bondad,
en su diaria tarea resplandecen
infinita su dadiva ideal.
Energia que modelas paisajes
insurgente y activo soñar,
un humano concepto sostiene
tu cultura de ser integral.
(CORO)
Mueve al hombre tu fe constructiva
se oye el ritmo de su despertar,
sinfonia de las urbes fabriles
alma agreste de un himno rural.
Corazon valeroso y ardiente
que edificas baluarte de paz
solidaria su accion con tus filas
vive el pueblo tu hermosa verdad.
```

(Figura 10)

¹⁰ Solución del ejercicio no. 4.

¹¹ Imagen del archivo .txt del himno del politécnico.

- 9) Se carga el archivo a la consola y se comprueba usando el comando ls (Véase Figura 11)¹².

```
Loading...

Welcome to JS/Linux (i586)

Use 'vlogin username' to connect to your account.
You can create a new account at https://vfsync.org/signup .
Use 'export_file filename' to export a file to your computer.
Imported files are written to the home directory.

localhost:~# ls
bench.py      hello.c      hello.js     himno_ipn.txt  readme.txt
localhost:~#
```

(Figura 11)

- 10) Se utiliza el comando less junto al nombre del archivo y su extensión como: less himno_ipn.txt, esto nos permitirá visualizar el himno del politécnico en la consola (Véase Figura 12)¹³.

```
Guinda y blanco, indomita almena
que defiende tu ardor juvenil,
oriflama en las lides gallardas
en tus manos triunfal bandern.
(CORO)
Tus brigadas de nitida albura
ciencia augusta, saber bondad,
en su diaria tarea resplandecen
infinita su dadiva ideal.
Energia que modelas paisajes
insurgente y activo soar,
un humano concepto sostiene
tu cultura de ser integral.
(CORO)
Mueve al hombre tu fe constructiva
se oye el ritmo de su despertar,
sinfonia de las urbes fabriles
alma agreste de un himno rural.
Corazon valeroso y ardiente
que edificas baluarte de paz
solidaria su accin con tus filas
vive el pueblo tu hermosa verdad.
~
(END)
```

(Figura 12)

¹² Muestra del archivo ingresado a la consola por el comando ls.

¹³ Muestra del contenido del archivo .txt por el comando less.

- 11) Se utilizó -E para introducir varios patrones de búsqueda en la terminal así como -v para mostrar las líneas que no coinciden con el patron buscado (Véase Figura 13)¹⁴.

El patrón es: -E -v [aeiou]{2} himno_ipn.txt

```
localhost:~# grep -E -v [aeiou]{2} himno_ipn.txt
una estrella te puso en las manos,
¡que no apague su limpio fulgor!

(CORO)

Su libertad
nace la luz;
nos convoca tu voz, Politecnico,
nos conduce tu amor, juventud.

En dinamico anhelo conjugas
es la tecnica impulso motriz.

oriflama en las lides gallardas

(CORO)

Tus brigadas de nitida albura
insurgente y activo soar,
tu cultura de ser integral.

(CORO)

se oye el ritmo de su despertar,
alma agreste de un himno rural.
```

(Figura 13)

- 12) Después se proporcionó un link para consultar un tutorial de comandos para utilizar el regex de Python (Véase Figura 14)¹⁵.

<https://relopezbriega.github.io/blog/2015/07/19/expresiones-regulares-con-python/>

(Figura 14)

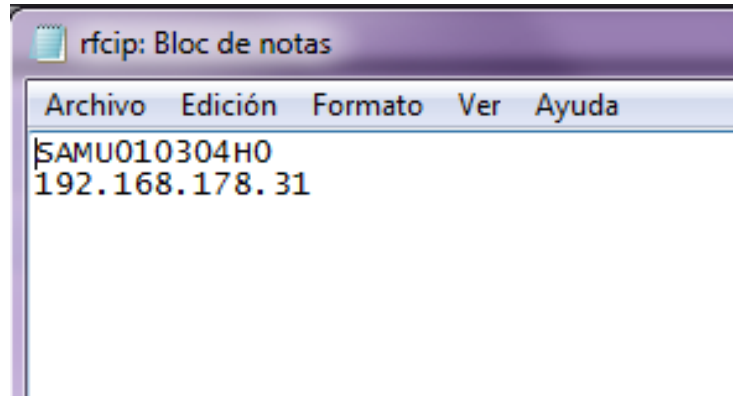
¹⁴ Solución a la segunda parte de la práctica haciendo uso de grep.

¹⁵ Enlace para una guía de los comandos y el uso de re en Python.

- 13) Se tienen que comprobar RFC y IPV4 con las expresiones regulares tanto como en Python y en grep de UNIX

Mi programa en Python lo hice leyendo un archivo externo de tipo texto (Véase Figura 15)¹⁶, y empleando las funciones de la extensión que importe “re” determine mediante expresiones regulares si coincidía con el contenido que estaba en el archivo tipo texto.

- 14) El archivo tipo texto es el siguiente:



(Figura 15)

- 15) El archivo tipo texto debe almacenars en la misma ubicación donde se encuentre tu programa de extensión .py (Véase Figura 16)¹⁷.

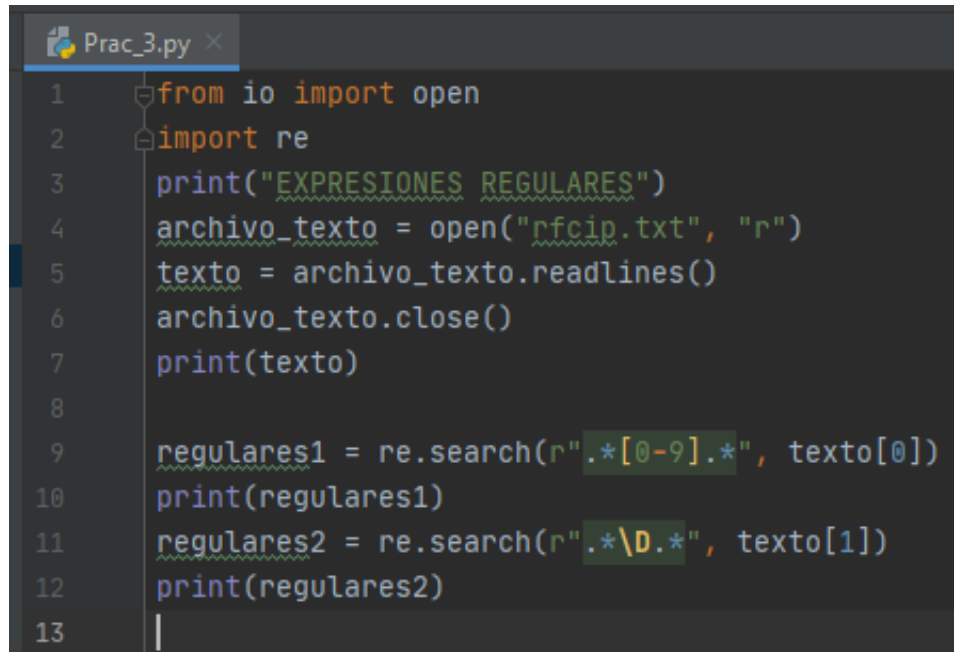
.idea	13/11/2020 10:46 a...	Carpeta de archivos	
Include	13/11/2020 07:58 a...	Carpeta de archivos	
Lib	13/11/2020 07:58 a...	Carpeta de archivos	
Scripts	13/11/2020 08:02 a...	Carpeta de archivos	
Prac_3	13/11/2020 10:46 a...	Archivo PY	1 KB
pyvenv.cfg	13/11/2020 07:58 a...	Archivo CFG	1 KB
rfcip	13/11/2020 08:46 a...	Documento de tex...	1 KB

(Figura 16)

¹⁶ Imagen del archivo de texto donde guardé el RFC y la IPV4.

¹⁷ Ubicación del archivo .py así como el archivo .txt.

16) El código del programa es el siguiente (Véase Figura 17)¹⁸:



```
1 from io import open
2 import re
3 print("EXPRESIONES REGULARES")
4 archivo_texto = open("rfcip.txt", "r")
5 texto = archivo_texto.readlines()
6 archivo_texto.close()
7 print(texto)
8
9 regulares1 = re.search(r".*[0-9].*", texto[0])
10 print(regulares1)
11 regulares2 = re.search(r".*\D.*", texto[1])
12 print(regulares2)
13 |
```

(Figura 17)

17) Al hacer la ejecución nos indica la función de re que hubo coincidencia con la línea que está leyendo y es que la función readlines() se encarga de leer línea por línea lo que esta en el documento(Véase Figura 18)¹⁹.



```
"F:\Teoria Computacional\Programas\P3\Scripts\python.exe" "F:/Teoria Computacional/Programas/P3/Prac_3.py"
EXPRESIONES REGULARES
['SAMU010304H0\n', '192.168.178.31']
<re.Match object; span=(0, 12), match='SAMU010304H0'>
<re.Match object; span=(0, 14), match='192.168.178.31'>

Process finished with exit code 0
```

(Figura 18)

¹⁸ Código en Python para determinar si es la expresión regular del RFC y el IPv4.

¹⁹ Salida en consola diciendo que la función re encontró coincidencia con el RFC y el IPv4.

18) Para hacerlo en grep de LINUX se tiene que cargar el archivo .txt a la consola de la página web (Véase Figura 19)²⁰.

(Figura 19)

19) Después abrimos el contenido del archivo de texto con el comando `less` (Véase Figura 20)²¹.

(Figura 20)

20) Por expresiones regulares hacemos la búsqueda con grep para que tengamos de resultado ambas líneas de texto (Véase Figura 21)²².

(Figura 21)

²⁰ Entrada con el comando ls para saber si se ingresó el archivo rfcip.txt.

²¹ Visualización del contenido de rfcip.txt por el comando less.

²² Salida al ingresar la expresión regular para el contenido del archivo de texto.

Conclusiones:

En conclusión en esta práctica se hizo uso de grep y nos ayudó a encontrar con expresiones regulares cadenas específicas de un archivo, así como en Python se ocupó la función importada re para hacer uso de todas las funciones para manejar conjuntos de cadenas así como tuplas y diccionarios, se empleó el manejo de archivos externos con éxito, a su vez se utilizaron algunos comandos de Linux y sirvió para conocer la diferencia entre la operación de expresiones regulares en Linux y en Python.

Bibliografía:

- *Practique la teoría de autómatas y lenguajes formales*, Leonardo Alonso Hernández Rodríguez, 2010.
- *Python para todos*, Raúl González Duque, 2007.
- *Introducción a la programación en Python*, Andrés Marzal, Isabel García, 2010.
- *Teoría de autómatas y lenguajes formales*, Serafín Moral.
- *Teorías de Autómatas y lenguajes formales*, Elena Jurado Málaga, 2008.
- *Teoría de la computación; Lenguajes formales, autómatas y complejidad*, J. Glenn Brookshear.
- *Teoría de autómatas, lenguajes y computación*, Jeffrey Ullman, 2007.