

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

TEORÍA COMPUTACIONAL

PRÁCTICA 04

ALUMNO: HERNÁNDEZ CASTELLANOS CÉSAR URIEL

PROFESOR: ROSAS TRIGUEROS JORGE LUIS

FECHA DE REALIZACIÓN: 7 DE SEPTIEMBRE DEL 2017

FECHA DE ENTREGA: 10 DE SEPTIEMBRE DEL 2017

Marco teórico.

Una expresión regular es un modelo con el que el motor de expresiones regulares intenta buscar una coincidencia en el texto de entrada. Un modelo consta de uno o más literales de carácter, operadores o estructuras.

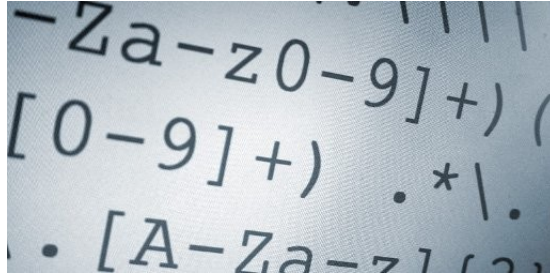


Imagen 1.0 Expresión regular.

Componentes de las Expresiones Regulares

Las expresiones regulares son un mini lenguaje en sí mismo, por lo que para poder utilizarlas eficientemente primero debemos entender los componentes de su sintaxis; ellos son:

Literales: Cualquier carácter se encuentra a sí mismo, a menos que se trate de un metacarácter con significado especial. Una serie de caracteres encuentra esa misma serie en el texto de entrada, por lo tanto la plantilla "raul" encontrará todas las apariciones de "raul" en el texto que procesamos.

Secuencias de escape: La sintaxis de las expresiones regulares nos permite utilizar las secuencias de escape que ya conocemos de otros lenguajes de programación para esos casos especiales como ser finales de línea, tabs, barras diagonales, etc.

Clases de caracteres: Se pueden especificar clases de caracteres encerrando una lista de caracteres entre corchetes [], la que encontrará uno cualquiera de los caracteres de la lista. Si el primer símbolo después del "[" es "^", la clase encuentra cualquier carácter que no está en la lista.

Metacaracteres: Los metacaracteres son caracteres especiales que son la esencia de las expresiones regulares. Como son sumamente importantes para entender la sintaxis de las expresiones regulares y existen diferentes tipos, voy a dedicar una sección a explicarlos un poco más en detalle.

Expresiones Regulares. Módulo re

Módulo re

Para utilizar Expresiones Regulares, Python provee el módulo re. Importando este módulo podemos crear objetos de tipo patrón y generar objetos tipo matcher, que son los que contienen la información de la coincidencia del patrón en la cadena.

Creando un patrón

Para crear un objeto patrón, se debe importar el módulo re y utilizamos la función compile:

```
import re

patron = re.compile('a[3-5]+') # coincide con una letra, seguida de al menos 1
                                # dígito entre 3 y 5
```

Buscar el patrón en la cadena

Para buscar un patrón en una cadena, Python provee los métodos search y match. La diferencia entre ambos es que, mientras search busca en la cadena alguna ocurrencia del patrón, match devuelve None si la ocurrencia no se da al principio de la cadena:

```
cadena = 'a44453'
patron.match(cadena) # <_sre.SRE_Match object at 0x02303BF0>
patron.search(cadena) # <_sre.SRE_Match object at 0x02303C28>
cadena = 'ba3455' # la coincidencia no está al principio!
patron.search(cadena) # <_sre.SRE_Match object at 0x02303BF0>
print patron.match(cadena) # None
```

Reemplazo de cadenas

Similar a la combinación search + expand, existe el método [sub](#) cuya función es encontrar todas las coincidencias de un patrón y sustituirlas por una cadena. Este recibe dos parámetros: el primero es la cadena con la que se sustituirá el patrón y el segundo es la cadena sobre la que queremos aplicar la sustitución.

```
patron.sub("X", 'a455 a333b435') # sustituye todas las ocurrencias por X 'X XX'
patron.sub("LETRA(\g<1>), NUMERO(\g<2>)", 'a455 a333b435') # El reemplazo depende
de lo que se capture LETRA(a), NUMERO(455) LETRA(a), NUMERO(333)LETRA(b),
NUMERO(435)'
```

Material y equipo.

El material utilizado en la práctica es el siguiente:

Herramientas de software:

- * Oracle VM VirtualBox
- * Ubuntu 17.04
- * Python 3.6.2
- * eric6 Web Browser (QtWebKit)
- * Online regez texter and debugger.

Herramientas de hardware:

- * Computadora personal

Desarrollo de la práctica.

La práctica número cuatro de la unidad de aprendizaje teoría computacional, consistió básicamente en practicas nuestras habilidades al formar expresiones regulares.

La primera sección de la práctica se encontradaba fragmentada en cuatro ejercicios, los cuales iban aumentando gradualmente de dificultad, el propósito de los ejercicios, era el de encontrar el patrón que representaba la parte positiva, la cual debería de ser iluminada de rojo para proseguir con el siguiente nivel.

La segunda parte de la práctica consistió el elaborar diez ejercicios de expresiones regulares, con la particularidad que se usó la librería re, para poder trabajar con expresiones regulares en Python.

Las dificultades en la práctica fue principalmente lo intimidante que puede llegar a ser las expresiones regulares a primera impresión, esto se solucionó investigando en diferentes fuentes los diferentes comandos que nos brindan, para poder generar nuestros propios patrones, también una de las dificultades fue el gran periodo de tiempo invertido en los ejercicios, en los cuales algunas veces se avanzaba de forma fluida y en otras pasaban las horas sin encontrar solución alguna.

A continuación se presentan los resultados obtenidos en la práctica:

Diagramas, gráficas y pantallas.

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
spot
spate
slap two
respite

pt
Pot
peat
part

Imagen 1.1 - Primer ejercicio.

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

rap them
tapeth
apth
wrap/try
sap tray
87ap9th
apothecary

aleht
happy them
tarpth
Apt
peth
tarreth
ddapdg
apples
shape the

Imagen 1.2 Segundo Ejercicio.

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

affgking fgok
rafgkahe a fgk
bafghk affgm
baffgkit afffhk
affgking fgok
rafgkahe afg.K
bafghk aff gm
baffg kit afffhgk

Imagen 1.3 Tercer ejercicio.

Exercise 4: Finding sentence breaks

Finding where one sentence ends and another begins is trickier than might be imagined. 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

assumes word senses. Within
does the clustering. In the
but when? It was hard to tell
he arrive." After she had
mess! He did not let it
it wasn't hers!" She replied
always thought so.) Then

Negative

in the U.S.A., people often
John?", he often thought, but
weighed 17.5 grams
well ... they'd better not
A.I. has long been a very
like that", he thought
but W. G. Grace never had much

Imagen 1.4 Cuarto Ejercicio.

```
uriel@Uriel-PC:~/Escritorio$ cd teoriac
uriel@Uriel-PC:~/Escritorio/teoriac$ ls
Programa01 Programa03 Programa05 Programa07 Programa09
Programa02 Programa04 Programa06 Programa08 Programa10
uriel@Uriel-PC:~/Escritorio/teoriac$ cd Programa01
uriel@Uriel-PC:~/Escritorio/teoriac/Programa01$ python regexpy.py
['aaksdsd', 'aaoaksb', 'aalsdlskl', 'aabasasab', 'aabbbbb', 'aabsa', 'aaidjsdisjidjsijidjsijdsisdbbb']
uriel@Uriel-PC:~/Escritorio/teoriac/Programa01$
```

Imagen 1.5 Programa número uno.

- Cadenas que empiecen con dos a's y tengan cero o más b's.

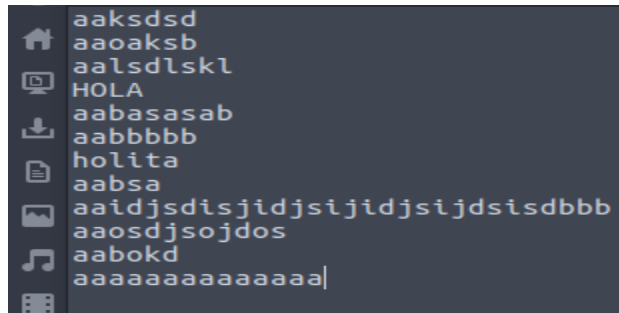
```
Abrir ▾ [F1]
aaksdsd
aaoaksb
aalsdlskl
HOLA
aabasasab
aabbbbb
holita
aabsa
aaidjsdisjidjsijidjsijdsisdbbb|
```

Imagen 1.6 Contenido del fichero.

```
['aaksdsd', 'aaoaksb', 'aalsdlskl', 'aabasasab', 'aabbbbb', 'aabsa', 'aaidjsdisjidjsijidjsijdsisdbbb']
uriel@Uriel-PC:~/Escritorio/teoriac/Programa01$ cd ..
uriel@Uriel-PC:~/Escritorio/teoriac$ cd Programa02
uriel@Uriel-PC:~/Escritorio/teoriac/Programa02$ python regexpy.py
['aaoaksb', 'aabasasab', 'aabbbbb', 'aabsa', 'aaidjsdisjidjsijidjsijdsisdbbb', 'aabokd']
uriel@Uriel-PC:~/Escritorio/teoriac/Programa02$
```

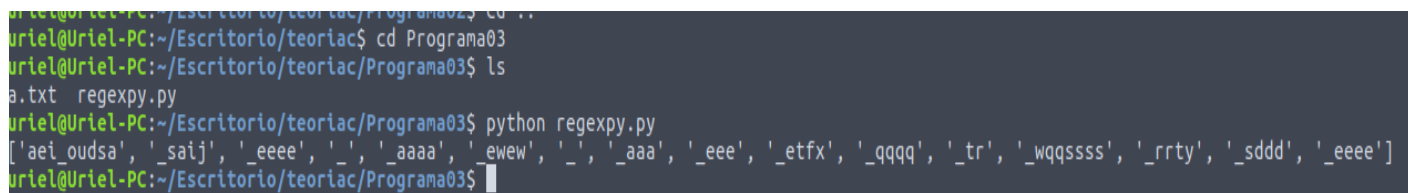
Imagen 1.7 Programa número dos.

- Cadenas que empiecen con dos a's y tengan una o más b's.



```
aaksdsd
aaoaksb
aalsdlskl
HOLA
aabasasab
aabbbbb
holita
aabsa
aaidjsdisjidjsijidjsijdsisdbbb
aaosdjsojdos
aabokd
aaaaaaaaaaaaa|
```

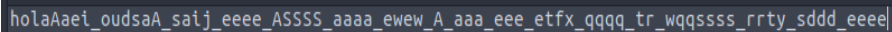
Imagen 1.8 Contenido del fichero.



```
Uriel@Uriel-PC:~/Escritorio/teoriac/Programa03$ cd ..
Uriel@Uriel-PC:~/Escritorio/teoriac$ cd Programa03
Uriel@Uriel-PC:~/Escritorio/teoriac/Programa03$ ls
a.txt  regexpy.py
Uriel@Uriel-PC:~/Escritorio/teoriac/Programa03$ python regexpy.py
['aei_oudsa', '_saij', '_eeee', '_', '_aaaa', '_ewew', '_', '_aaa', '_eee', '_etfx', '_qqqq', '_tr', '_wqqssss', '_rrty', '_sddd', '_eeee']
Uriel@Uriel-PC:~/Escritorio/teoriac/Programa03$
```

Imagen 1.9 Programa número tres.

- Cadenas en donde dos subcadenas de letras minúsculas estén unidas por un guión bajo.

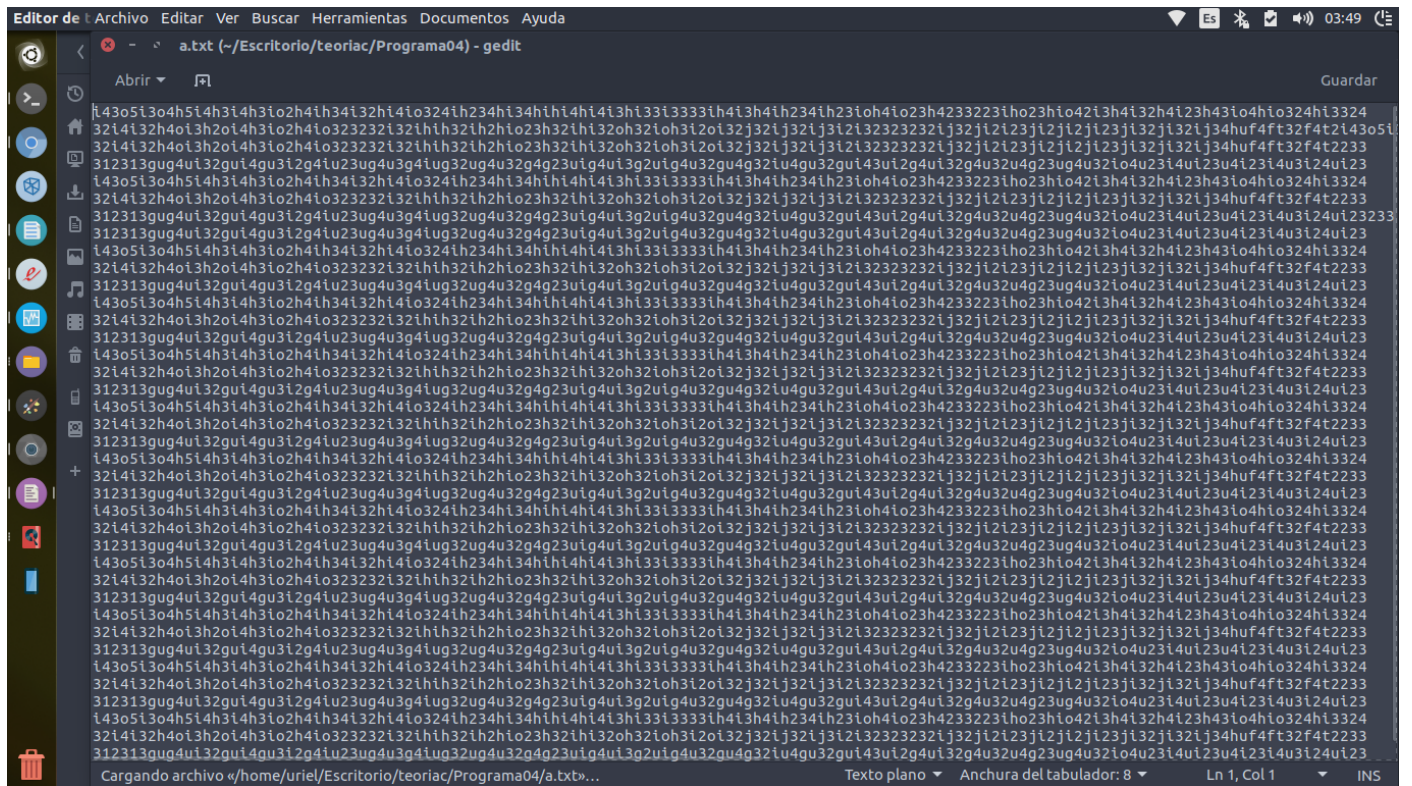


```
holaAaei_oudsaA_saij_eeee_ASSSS_aaaa_ewew_A_aaa_eee_etfx_qqqq_tr_wqqssss_rrty_sddd_eeee|
```

Imagen 2.0 Contenido del fichero.

[illegible]

Imagen 2.1 Programa número cuatro.



The image shows a screenshot of a text editor window titled "Editor de Archivo" with a menu bar including "Archivo", "Editar", "Ver", "Buscar", "Herramientas", "Documentos", and "Ayuda". The file name in the title bar is "a.txt (~/Escritorio/teoriac/Programa04) - gedit". The editor displays a large block of Base64-encoded text, which is a repeating pattern of the same 100-character string. The status bar at the bottom indicates "Cargando archivo «/home/uriel/Escritorio/teoriac/Programa04/a.txt»...", "Texto plano", "Anchura del tabulador: 8", "Ln 1, Col 1", and "INS".

```
l43o5i3o4h5i4h3i4h3i2o4h4i3h4i32hi4i3o324ih234hi34hi4hi4i4i3hi33i3333ih4i3h4ih234ih23ioh4io23h4233223i2ho23hi4o2i3h4i32h4i23h43i2o4hio324hi3324
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2i43o5i
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2233
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
l43o5i3o4h5i4h3i4h3i2o4h4i3h4i32hi4i3o324ih234hi34hi4hi4i4i3hi33i3333ih4i3h4ih234ih23ioh4io23h4233223i2ho23hi4o2i3h4i32h4i23h43i2o4hio324hi3324
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2233
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
l43o5i3o4h5i4h3i4h3i2o4h4i3h4i32hi4i3o324ih234hi34hi4hi4i4i3hi33i3333ih4i3h4ih234ih23ioh4io23h4233223i2ho23hi4o2i3h4i32h4i23h43i2o4hio324hi3324
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2233
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
l43o5i3o4h5i4h3i4h3i2o4h4i3h4i32hi4i3o324ih234hi34hi4hi4i4i3hi33i3333ih4i3h4ih234ih23ioh4io23h4233223i2ho23hi4o2i3h4i32h4i23h43i2o4hio324hi3324
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2233
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
l43o5i3o4h5i4h3i4h3i2o4h4i3h4i32hi4i3o324ih234hi34hi4hi4i4i3hi33i3333ih4i3h4ih234ih23ioh4io23h4233223i2ho23hi4o2i3h4i32h4i23h43i2o4hio324hi3324
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2233
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
l43o5i3o4h5i4h3i4h3i2o4h4i3h4i32hi4i3o324ih234hi34hi4hi4i4i3hi33i3333ih4i3h4ih234ih23ioh4io23h4233223i2ho23hi4o2i3h4i32h4i23h43i2o4hio324hi3324
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2233
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
l43o5i3o4h5i4h3i4h3i2o4h4i3h4i32hi4i3o324ih234hi34hi4hi4i4i3hi33i3333ih4i3h4ih234ih23ioh4io23h4233223i2ho23hi4o2i3h4i32h4i23h43i2o4hio324hi3324
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2233
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
l43o5i3o4h5i4h3i4h3i2o4h4i3h4i32hi4i3o324ih234hi34hi4hi4i4i3hi33i3333ih4i3h4ih234ih23ioh4io23h4233223i2ho23hi4o2i3h4i32h4i23h43i2o4hio324hi3324
32i4i32h4oi3h2oi4h3io2h4io323232i32ihh32ih2hio23h32ih32oh32ioh3i2oi32j32ij32ij3j12i32323232ij32ji2i23ji2j2ij2j123ji32ij32ij34huf4ft32f4t2233
312313gug4ui32gui4gu3i2g4i23ug4u3g4iug32ug4u32g4g23uig4ui3g2uig4u32gu4g32i4u4gu32gu143ui2g4ui32g4u32u4g23ug4u32i4ou23i4ui23u4i23i4u3i24ui23
```

Imagen 2.2 Contenido del fichero.

Expression

share

save

flags

/(p[^p\s]*){2}/g

3 matches

Text

pamdmsapele

pandario

pandapaís

pandaxd

país panda

Imagen 2.3 Programa número cinco.

```

l Dr y le preguntó qué había que hacer para hacerse Dr. El Dr de broma, le contestó que debería de comprar un abecedario, encargar un par de tra
jes y poner un letrero que dijera "Dr Sabelotodo". Después de esto o, sólo tendría que esperar a que los clientes fuesen llegando.

El pobre leñador no entendió la broma y vendió su mula y con el dinero compró un abecedario, dos bonos trajes y una bata blanca de Dr, además de
poner en la puerta de su casa un cartel como le había indicado el Dr. Todo esto se los pecó a su mujer y le dijo que era para ganar mucho diner
o.

Habían pasado pocos días cuando un señor llegó preguntando por el Dr Sabelotodo. Le dijo al leñador que le habían robado todo su dinero y este l
e aseguro que le explicaría la forma en que podría recuperarlo. El cliente le invitó a comer a su mansión y a su mujer también. Una vez en la lu
josa casa, apareció un criado con el primer plato, dispuesto a servirlo y el leñador se puso tan contento que le dijo su mujer e que llegaba el
primero, refiriéndose al primer plato, pero el criado que era un ladrón se dio por aludido y avisó a los otros, advirtiéndole que el Dr ya sabía l
a verdad, que ellos habían robado dinero. Al aparecer el segundo criado pasó lo mismo y volvió la cocina muy asustado. Lo mismo sucedió con el t
ercer criado. El amo pidió al Dr Sabelotodo que adivinar a lo que cuarto criado traía una fuente. Y como eran cangrejos lo que había la fuente,
el criado llamó al falso Dr a la cocina y los ladrones le confesaron que eran los autores del robo y dijeron que estaban arrepentidos y pensaban
devolver el dinero.

Volvió Cangrejo al comedor y le dijo al amo que iba a leer en su abecedario donde estaba dinero robado. Sacó su libro, revisó algunas de las pág
inas y por fin dijo al caballero el sitio exacto donde estaba su dinero. Como hacer todo, el Dr Sabelotodo recibió una gran recompensa, además d
e otra recompensa por parte de los criados y así fue como el pobre leñador se hizo rico y nunca más tuvo necesidad de ir al bosque a cortar leña
. Vivió muy feliz con su mujer.

urtel@Urtel-PC:~/Escritorio/teoriac/Programa06$ grep "Dr" a.txt
Erase una vez un buen leñador que se llamaba Cangrejo que vivía muy pobremente. Un buen día, Cangrejo pareció un fuerte dolor de espalda y como
no podía mover el hacha para trabajar, fue a la consulta de un Dr. Cuando llegó a casa del Dr observó que era muy lujosa y nada tenía que ver co
n la pobre castita en donde vivía el, que más bien era una cabaña. El leñador de inmediato pensó que si se hacia Dr podría vivir ricamente, que e
l Dr y le preguntó qué había que hacer para hacerse Dr. El Dr de broma, le contestó que debería de comprar un abecedario, encargar un par de tra
jes y poner un letrero que dijera "Dr Sabelotodo". Después de esto o, sólo tendría que esperar a que los clientes fuesen llegando.

El pobre leñador no entendió la broma y vendió su mula y con el dinero compró un abecedario, dos bonos trajes y una bata blanca de Dr, además de
poner en la puerta de su casa un cartel como le había indicado el Dr. Todo esto se los pecó a su mujer y le dijo que era para ganar mucho diner
o.

Habían pasado pocos días cuando un señor llegó preguntando por el Dr Sabelotodo. Le dijo al leñador que le habían robado todo su dinero y este l
e aseguro que le explicaría la forma en que podría recuperarlo. El cliente le invitó a comer a su mansión y a su mujer también. Una vez en la lu
josa casa, apareció un criado con el primer plato, dispuesto a servirlo y el leñador se puso tan contento que le dijo su mujer e que llegaba el
primero, refiriéndose al primer plato, pero el criado que era un ladrón se dio por aludido y avisó a los otros, advirtiéndole que el Dr ya sabía l
a verdad, que ellos habían robado dinero. Al aparecer el segundo criado pasó lo mismo y volvió la cocina muy asustado. Lo mismo sucedió con el t
ercer criado. El amo pidió al Dr Sabelotodo que adivinar a lo que cuarto criado traía una fuente. Y como eran cangrejos lo que había la fuente,
el criado llamó al falso Dr a la cocina y los ladrones le confesaron que eran los autores del robo y dijeron que estaban arrepentidos y pensaban
devolver el dinero.

Volvió Cangrejo al comedor y le dijo al amo que iba a leer en su abecedario donde estaba dinero robado. Sacó su libro, revisó algunas de las pág
inas y por fin dijo al caballero el sitio exacto donde estaba su dinero. Como hacer todo, el Dr Sabelotodo recibió una gran recompensa, además d
e otra recompensa por parte de los criados y así fue como el pobre leñador se hizo rico y nunca más tuvo necesidad de ir al bosque a cortar leña
. Vivió muy feliz con su mujer.
urtel@Urtel-PC:~/Escritorio/teoriac/Programa06$

```

Imagen 2.3 Programa número seis.

```
a.txt prog2.py
Uriel@Uriel-PC:~/Escritorio/teoriac/Programa07$ python prog2.py
Érase; una; vez; un; buen; leñador; que; se; llamaba; Cangrejo; que; vivía; muy; pobremente;; Un; buen; día;; Cangrejo; pareció; un; fuerte; dolor; de; espalda; y; como;
no; podía; mover; el; hacha; para; trabajar;; fue; a; la; consulta; de; un; Doctor;; Cuando; llegó; a; casa; del; Doctor; observó; que; era; muy; lujosa; y; nada; tenía; qu
e; ver; con; la; pobre; casita; en; donde; vivía; el;; que; más; bien; era; una; cabaña;; El; leñador; de; inmediato; pensó; que; si; se; hacía; Doctor; podría; vivir; rica
mente;; que; el; Doctor; y; le; preguntó; qué; había; que; hacer; para; hacerse; Doctor;; El; Doctor; de; broma;; le; contestó; que; debería; de; comprar; un; abecedario
;; encargar; un; par; de; trajes; y; poner; un; letrero; que; dijera; "Doctor; Sabelotodo"; Después; de; esto; o;; sólo; tendría; que; esperar; a; que; los; clientes; fu
esen; llegando;; El; pobre; leñador; no; entendió; la; broma; y; vendió; su; mula; y; con; el; dinero; compró; un; abecedario;; dos; bonos; trajes; y; una; bata; blanca;
de; Doctor;; además; de; poner; en; la; puerta; de; su; casa; un; cartel; como; le; había; indicado; el; Doctor;; Todo; esto; se; los; peco; a; su; mujer; y; le; dijo; que; er
a; para; ganar; mucho; dinero;; Habían; pasado; pocos; días; cuando; un; señor; llegó; preguntando; por; el; Doctor; Sabelotodo;; Le; dijo; al; leñador; que; le; había
n; robado; todo; su; dinero; y; este; le; aseguró; que; le; explicaría; la; forma; en; que; podría; recuperarlo;; El; cliente; le; invitó; a; comer; a; su; mansión; y; a; su
mujer; también;; Una; vez; en; la; lujosa; casa; apareció; un; criado; con; el; primer; plato;; dispuesto; a; servirlo; y; el; leñador; se; puso; tan; contento; que; le
; dijo; su; mujer; e; que; llegaba; el; primero;; refiriéndose; al; primer; plato;; pero; el; criado; que; era; un; ladrón; se; dio; por; aludido; y; avisó; a; los; otros;;
advirtiéndolo; que; el; Doctor; ya; sabía; la; verdad; que; ellos; habían; robado; dinero;; Al; aparecer; el; segundo; criado; pasó; lo; mismo; y; volvió; la; cocina; muy
; asustado;; Lo; mismo; sucedió; con; el; tercer; criado;; El; amo; pidió; al; Doctor; Sabelotodo; que; adivinar; a; lo; que; cuarto; criado; traía; una; fuente;; Y; como
; eran; cangrejos; lo; que; había; la; fuente; el; criado; llamó; al; falso; Doctor; a; la; cocina; y; los; ladrones; le; confesaron; que; eran; los; autores; del; robo; y
; dijeron; que; estaban; arrepentidos; y; pensaban; devolver; el; dinero;; Volvió; Cangrejo; al; comedor; y; le; dijo; al; amo; que; iba; a; leer; en; su; abecedario; do
nde; estaba; dinero; robado;; Sacó; su; libro;; revisó; algunas; de; las; páginas; y; por; fin; dijo; al; caballero; el; sitio; exacto; donde; estaba; su; dinero;; Como;
hacer; todo;; el; Doctor; Sabelotodo; recibió; una; gran; recompensa; además; de; otra; recompensa; por; parte; de; los; criados; y; así; fue; como; el; pobre; leñador
; se; hizo; rico; y; nunca; más; tuvo; necesidad; de; ir; al; bosque; a; cortar; leña;; Vivió; muy; feliz; con; su; mujer;;
Uriel@Uriel-PC:~/Escritorio/teoriac/Programa07$
```

Imagen 2.4 Programa número siete.

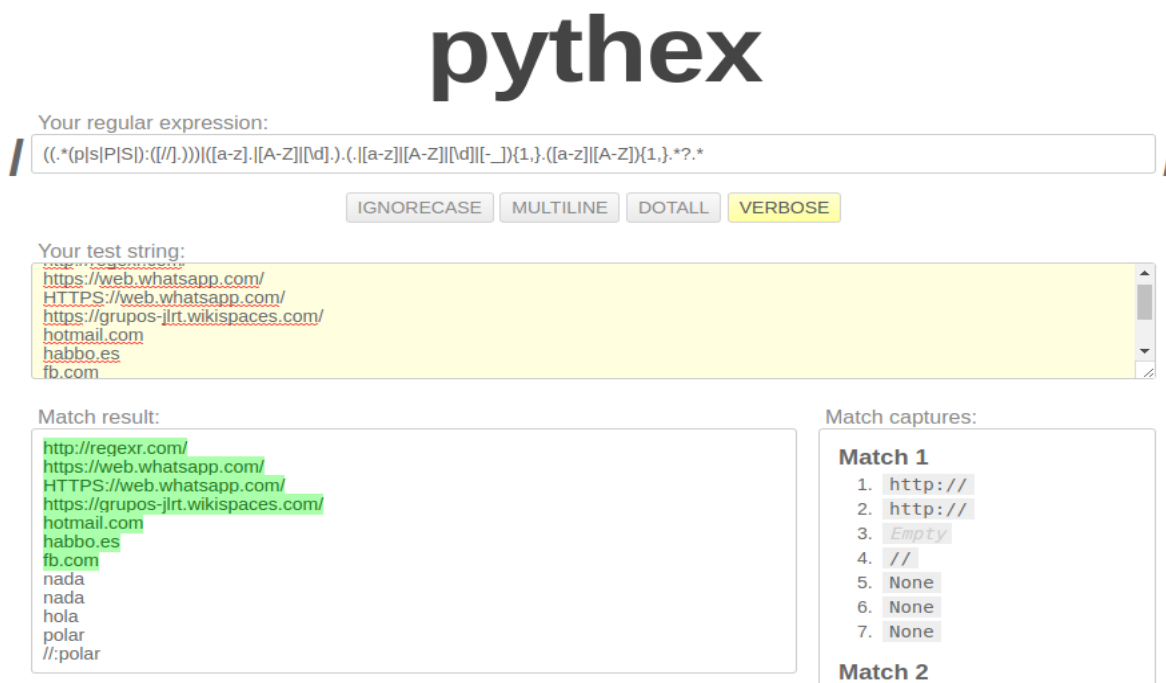


Imagen 2.5 Programa número ocho.

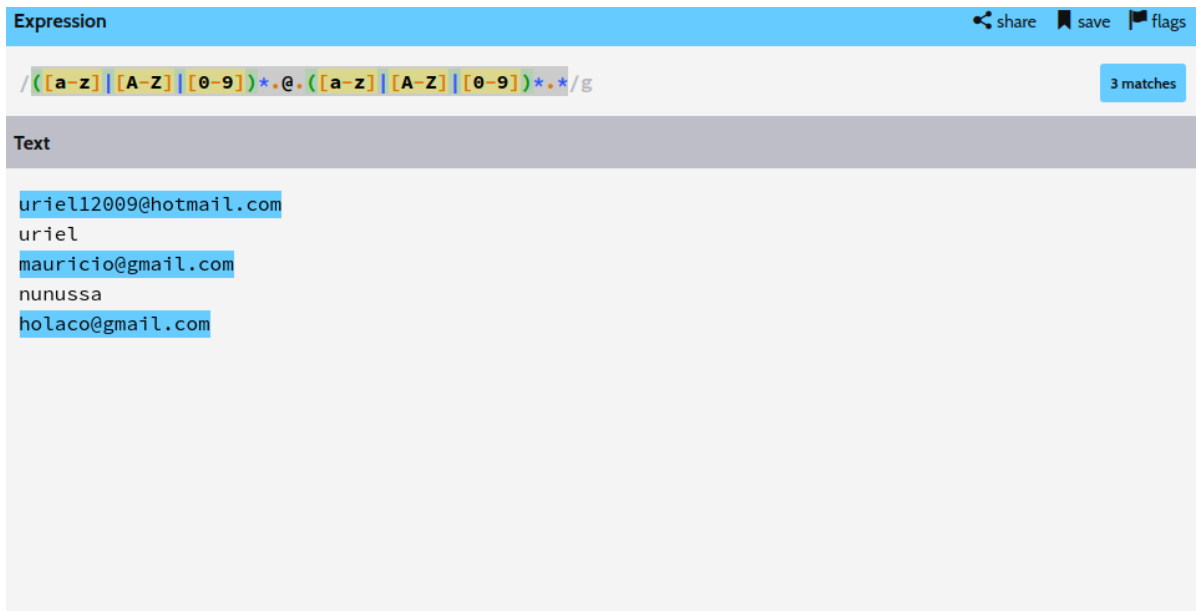


Imagen 2.6 Programa número nueve.

```
uriel@Uriel-PC: ~/Escritorio
uriel@Uriel-PC:~$ cd Escritorio
uriel@Uriel-PC:~/Escritorio$ python nuevo.py
- - - - -
uriel@Uriel-PC:~/Escritorio$
```

Imagen 2.7 Programa número diez.

Expresiones regulares usadas:

1-. aa.*b{0,}?.*
2-. aa.*b{1,}?.*
3-. [a-z]*_[a-z]*
4-. [0-9]{1,3}
5-. (p[^p\s]*){2}
6-. Doctor
7-. [\s, .]
8-. ((.*(p|s|P|S)):([/\].))|([a-z]|[A-Z]|[\d]).)(.|[a-z]|[A-Z]|
[\d]|[-_]){1,}.*?
9-. ([a-z]|[A-Z]|[0-9])*@.([a-z]|[A-Z]|[0-9]).*

Nota: El código de los programas se anexó en el comprimido de la práctica.

Conclusiones y recomendaciones.

Las expresiones regulares resultan una herramienta poderosa para un programador, ya que reduce el tiempo de desarrollo de manera considerable, además de contar con un código más compacto.

La resolución de las expresiones regulares no es única, lo que pude percatarme que para dar resolución a estos problemas la mejor técnica a emplear es la divide y vencerás.

Referencias

- [1]"Lenguaje de expresiones regulares - Referencia rápida", *Msdn.microsoft.com*, 2017. [Online]. Available: [https://msdn.microsoft.com/es-es/library/az24scfc\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/az24scfc(v=vs.110).aspx). [Accessed: 10- Sep- 2017].
- [2]"Regular expressions | Sketch Engine", *Sketchengine.co.uk*, 2017. [Online]. Available: <https://www.sketchengine.co.uk/user-guide/user-manual/concordance-introduction/regular-expressions/>. [Accessed: 10- Sep- 2017].
- [3]"Guía de expresiones regulares en Python", *platzi.com*, 2017. [Online]. Available: <https://platzi.com/blog/expresiones-regulares-python/>. [Accessed: 10- Sep- 2017].
- [4]S. Programacion en Castellano, "Expresiones regulares en Python", *Programación en Castellano.*, 2017. [Online]. Available: http://programacion.net/articulo/expresiones_regulares_en_python_1436. [Accessed: 10- Sep- 2017].
- [5]"Expresiones regulares en python - ChuWiki", *Chuwiki.chuidiang.org*, 2017. [Online]. Available: http://chuwiki.chuidiang.org/index.php?title=Expresiones_regulares_en_python. [Accessed: 10- Sep- 2017].
- [6]"Regex mediante ejemplos", *Pybonacci*, 2017. [Online]. Available: <https://pybonacci.es/2013/02/21/regex-mediante-ejemplos/>. [Accessed: 10- Sep- 2017].