Actividad 21. Uso de expresiones regulares

1. Introducción a las Expresiones Regulares

1.1. ¿Qué son las Expresiones Regulares?

Definición de Expresiones Regulares

Las expresiones regulares, comúnmente conocidas como **regex** o **regexp,** son secuencias de caracteres que forman un patrón de búsqueda. Este patrón se utiliza para llevar a cabo coincidencias dentro de cadenas de texto, permitiendo encontrar, validar, extraer o reemplazar subcadenas que cumplen con ciertos criterios.

Importancia y Uso en la Seguridad Informática

Las expresiones regulares son cruciales en la seguridad informática por varias razones:

- **Validación de Entradas**: Se utilizan para asegurarse de que los datos ingresados en formularios web o en sistemas críticos cumplan con el formato esperado, evitando así inyecciones de código malicioso.
- **Análisis de Logs**: Ayudan a buscar patrones específicos en archivos de registro (logs), lo cual es vital para la detección de intrusiones y el monitoreo de actividades sospechosas.
- **Filtrado y Sanitización de Datos**: Permiten limpiar datos de entrada y salida para prevenir ataques como la inyección SQL y la inyección de scripts (XSS).
- **Automatización de Tareas**: Facilitan la automatización de tareas repetitivas relacionadas con la manipulación de texto y la extracción de información relevante.

Ejemplos de Casos de Uso

- **Validación de Emails**: ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$
- **Búsqueda de Direcciones IP**: \b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b
- **Extracción de Fechas**: \b\d{2}/\d{2}/\d{4}\b

1.2. Conceptos Básicos

Metacaracteres

Los metacaracteres son caracteres con un significado especial dentro de una expresión regular:

- **.** : Coincide con cualquier carácter excepto un salto de línea.
- **^** : Coincide con el inicio de una línea.
- **$** : Coincide con el final de una línea.
- ***** : Coincide con cero o más repeticiones del carácter anterior.
- **+** : Coincide con una o más repeticiones del carácter anterior.
- **?** : Coincide con cero o una repetición del carácter anterior.
- **{}** : Especifica un número exacto de repeticiones del carácter anterior. Ejemplo: a{3} coincide con "aaa".
- **[]** : Define un conjunto de caracteres. Ejemplo: [abc] coincide con "a", "b" o "c".
- **()** : Define un grupo de caracteres. Utilizado también para capturar grupos.
- **|** : Actúa como un operador OR. Ejemplo: a|b coincide con "a" o "b".

Secuencias de Escape

Las secuencias de escape representan caracteres especiales:

- **\d** : Coincide con cualquier dígito (0-9).
- **\w** : Coincide con cualquier carácter alfanumérico (letras y números) y el guion bajo.
- **\s :** Coincide con cualquier carácter de espacio en blanco (espacios, tabulaciones, saltos de línea).

Grupos y Rangos

- **[abc]** : Coincide con cualquiera de los caracteres "a", "b" o "c".
- **[a-z]** : Coincide con cualquier letra minúscula.
- **[A-Z]** : Coincide con cualquier letra mayúscula.
- **[0-9]** : Coincide con cualquier dígito.

Ejemplos Simples

- **Coincidir con una dirección de email**: ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$
- **Coincidir con una URL**: https?://[^\s/$.?#].[^\s]*
- **Coincidir con una fecha en formato DD/MM/YYYY**: \b\d{2}/\d{2}/\d{4}\b
- **Coincidir con un número de teléfono**: \b\d{3}[-.]?\d{3}[-.]?\d{4}\b

2. Expresiones Regulares en Linux

2.1. Herramientas de Búsqueda en Linux

Linux ofrece una amplia variedad de herramientas que permiten utilizar expresiones regulares para buscar y manipular texto. Las más comunes y útiles en la administración y seguridad informática son grep, sed y awk.

2.1.1. grep (Global Regular Expression Print)

grep es una herramienta de búsqueda que permite buscar patrones en archivos de texto utilizando expresiones regulares.

Sintaxis Básica

```
grep [opciones] 'patrón' [archivo]
```

Opciones Comunes

- -i: Ignorar mayúsculas y minúsculas.
- -r: Buscar recursivamente en directorios.
- -l: Mostrar solo los nombres de los archivos que contienen el patrón.
- -n: Mostrar los números de línea donde se encontró el patrón.
- -v: Invertir la coincidencia, mostrar las líneas que no contienen el patrón.

Ejemplos

- Buscar una cadena específica en un archivo:

```
grep 'error' /var/log/syslog
```

- Buscar recursivamente en un directorio:
```
grep -r 'error' /var/log/
```
- Buscar líneas que no contienen una cadena:
```
grep -v 'error' /var/log/syslog
```

2.1.2. sed (Stream Editor)

sed es una herramienta de edición de flujo que permite realizar transformaciones en texto mediante el uso de expresiones regulares.

Sintaxis Básica

```
sed [opciones] 'script' [archivo]
```

Opciones Comunes

- -n: Suprimir la salida automática de la línea.
- -e: Permitir múltiples comandos de edición.
- -i: Editar los archivos en el lugar.

Comandos Básicos

- s/patrón/reemplazo/: Sustituir el patrón por el reemplazo.
- d: Eliminar líneas que coincidan con el patrón.
- p: Imprimir líneas que coincidan con el patrón.

Ejemplos

- Sustituir una cadena en un archivo:
  ```
  sed 's/error/ERROR/' /var/log/syslog
  ```
- Eliminar líneas que contienen una cadena:
  ```
  sed '/error/d' /var/log/syslog
  ```
- Imprimir líneas que contienen una cadena:
  ```
  sed -n '/error/p' /var/log/syslog
  ```

2.1.3. awk

awk es una potente herramienta de procesamiento de texto que permite extraer y manipular datos de archivos de texto estructurados.

Sintaxis Básica

```
awk 'patrón {acción}' [archivo]
```

Componentes

- patrón: Expresión regular para seleccionar líneas.
- acción: Acción a realizar sobre las líneas que coincidan con el patrón.

Ejemplos

- Imprimir líneas que contienen una cadena:
  ```
  awk '/error/ {print}' /var/log/syslog
  ```
- Imprimir la primera y tercera columna de un archivo:
  ```
  awk '{print $1, $3}' /var/log/syslog
  ```
- Sumar valores en una columna específica:
  ```
  awk '{sum += $2} END {print sum}' /var/log/syslog
  ```

2.2. Ejemplos Prácticos de Uso de Expresiones Regulares en Linux

2.2.1. Validación de Entradas

Las expresiones regulares pueden utilizarse para validar entradas de usuario en scripts de bash.

Ejemplo: Validar una dirección de correo electrónico:

```
#!/bin/bash
read -p "Introduce tu correo electrónico: " email
if [[ $email =~ ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]];
then
    echo "Correo válido"
else
    echo "Correo no válido"
fi
```

2.2.2. Búsqueda y Manipulación de Texto

Utilizando grep, sed y awk, se pueden realizar búsquedas avanzadas y manipulaciones en archivos de log y otros textos importantes para la seguridad informática.

Ejemplo: Buscar y resaltar una cadena en un archivo:

```
grep --color=auto 'error' /var/log/syslog
```

Ejemplo: Extraer direcciones IP de un archivo de log:

```
grep -oE '\b([0-9]{1,3}\.){3}[0-9]{1,3}\b' /var/log/syslog
```

Ejemplo: Sustituir direcciones IP en un archivo de log:

```
sed 's/\b([0-9]{1,3}\.){3}[0-9]{1,3}\b/[IP REDACTADA]/g'
/var/log/syslog
```

Ejemplo: Obtener estadísticas de uso de recursos en un archivo de log:

```
awk '/CPU/ {cpu_sum += $2; cpu_count++} END {print "Promedio de uso de
CPU:", cpu_sum/cpu_count}' /var/log/syslog
```

2.3. Consejos y Buenas Prácticas

- **Prueba tus expresiones regulares:** Antes de utilizarlas en scripts o comandos críticos, prueba tus expresiones regulares en ejemplos pequeños para asegurarte de que funcionan correctamente.
- **Utiliza opciones de simulación:** Muchas herramientas, como sed, tienen opciones que permiten simular cambios sin modificar los archivos originales. Utiliza estas opciones para evitar cambios no deseados.
- **Documenta tus expresiones:** Las expresiones regulares pueden ser difíciles de leer y entender. Añade comentarios y documentación a tus scripts para facilitar su mantenimiento.

2.4. Recursos Adicionales

- **Documentación Oficial:**
  - man grep
  - man sed
  - man awk
- **Tutoriales en Línea:**
  - Linux Regular Expressions Tutorial
  - GNU grep documentation
  - GNU sed documentation
  - GNU awk documentation

3. Expresiones Regulares en CMD

3.1. Introducción a las Herramientas de Búsqueda en CMD

En Windows CMD (Command Prompt), el uso de expresiones regulares no es tan nativo ni directo como en Linux. Sin embargo, hay algunas herramientas y comandos que permiten utilizar patrones de búsqueda avanzados. Algunas de estas herramientas incluyen findstr y powershell (utilizado desde CMD).

3.1.1. findstr

findstr es una herramienta de búsqueda de texto que permite buscar patrones específicos en archivos o en la salida de otros comandos. Aunque no es tan potente como grep en Linux, findstr soporta un conjunto básico de expresiones regulares.

Sintaxis Básica

```
findstr [opciones] "patrón" [archivo]
```

Opciones Comunes

- /I: Ignorar mayúsculas y minúsculas.
- /S: Buscar en archivos del directorio actual y todos los subdirectorios.
- /N: Mostrar los números de línea donde se encuentra el patrón.
- /V: Invertir la coincidencia, mostrar las líneas que no contienen el patrón.
- /R: Indicar que se está utilizando una expresión regular.

Ejemplos

- Buscar una cadena específica en un archivo:
```
findstr "error" archivo.txt
```
- Buscar recursivamente en un directorio:
```
findstr /S "error" *.txt
```
- Buscar líneas que contienen un patrón utilizando una expresión regular:
```
findstr /R "^ERROR.*[0-9]$" archivo.txt
```

3.2. Conceptos Básicos de Expresiones Regulares en findstr

findstr soporta un subconjunto limitado de expresiones regulares:

- .: Coincide con cualquier carácter excepto un salto de línea.
- *: Coincide con cero o más repeticiones del carácter anterior.
- ^: Coincide con el inicio de una línea.
- $: Coincide con el final de una línea.
- []: Coincide con cualquiera de los caracteres incluidos entre los corchetes.

Ejemplos Simples:

- Coincidir con cualquier línea que comienza con "ERROR":

```
findstr /R "^ERROR" archivo.txt
```

- Coincidir con cualquier línea que termina con un dígito:

```
findstr /R "[0-9]$" archivo.txt
```

3.3. Uso de PowerShell desde CMD

PowerShell, aunque es una herramienta diferente, puede ser invocada desde CMD y ofrece un soporte completo para expresiones regulares, mucho más potente que findstr.

Invocar PowerShell desde CMD:

```
powershell -Command "comando_de_powershell"
```

Ejemplos:

- Buscar y resaltar un patrón en un archivo utilizando PowerShell desde CMD:

```
powershell -Command "Select-String -Path 'archivo.txt' -Pattern 'error'"
```

- Filtrar la salida de un comando con expresiones regulares:

```
powershell -Command "Get-Content archivo.txt | Select-String -Pattern 'error'"
```

3.4. Ejemplos Prácticos de Uso de Expresiones Regulares en CMD

3.4.1. Búsqueda y Manipulación de Texto

Utilizando findstr y PowerShell, se pueden realizar búsquedas avanzadas y manipulaciones en archivos de log y otros textos importantes para la seguridad informática.

Ejemplo: Buscar y resaltar una cadena en un archivo:

```
findstr "error" archivo.txt
```

Ejemplo: Extraer direcciones IP de un archivo de log usando PowerShell desde CMD:

```
powershell -Command "Get-Content archivo.txt | Select-String -
Pattern '\b\d{1,3}(\.\d{1,3}){3}\b'"
```

Ejemplo: Sustituir direcciones IP en un archivo de log usando PowerShell desde CMD:

```
powershell -Command "(Get-Content archivo.txt) -replace
'\b\d{1,3}(\.\d{1,3}){3}\b', '[IP REDACTADA]' | Set-Content
archivo.txt"
```

3.5. Consejos y Buenas Prácticas

- **Prueba tus expresiones regulares:** Utiliza ejemplos pequeños para probar tus expresiones regulares antes de aplicarlas en scripts o comandos críticos.
- **Documenta tus scripts:** Las expresiones regulares pueden ser difíciles de leer y entender. Añade comentarios y documentación a tus scripts para facilitar su mantenimiento.
- **Utiliza PowerShell cuando sea posible:** Aunque CMD tiene algunas capacidades para expresiones regulares, PowerShell es mucho más potente y flexible. Considera utilizar PowerShell para tareas más complejas.

3.6. Recursos Adicionales

- **Documentación Oficial:**
  - Documentación de findstr
  - Documentación de PowerShell
- **Tutoriales en Línea:**
  - Using Findstr in the Windows Command Shell
  - PowerShell Regex Cheat Sheet

4. Expresiones Regulares en PowerShell

4.1. Introducción a PowerShell

PowerShell es una herramienta de automatización de tareas y un lenguaje de scripting de línea de comandos basado en .NET, que es mucho más potente y flexible que CMD. PowerShell tiene soporte completo para expresiones regulares, lo que lo hace ideal para la búsqueda y manipulación de texto, validación de entradas y más.

4.2. Uso de Expresiones Regulares en PowerShell

PowerShell utiliza el motor de expresiones regulares de .NET, lo que proporciona un soporte robusto para la creación y uso de expresiones regulares.

4.2.1. Comandos Básicos

En PowerShell, las expresiones regulares se pueden utilizar directamente con operadores y cmdlets como -match, -replace, Select-String y otros.

Operadores:

- -match: Coincide con una expresión regular.
- -notmatch: No coincide con una expresión regular.
- -replace: Sustituye un patrón de expresión regular.

Cmdlets:

- Select-String: Similar a grep en Linux, busca patrones en archivos o en la salida de otros comandos.

Ejemplos:

- Coincidir con una expresión regular:

```
"Hello123" -match "\d+"  # Devuelve True porque hay dígitos en la cadena
```

- No coincidir con una expresión regular:

```
"Hello" -notmatch "\d+"  # Devuelve True porque no hay dígitos en la cadena
```

- Reemplazar con una expresión regular:

```
"Hello123" -replace "\d+", "World"  # Devuelve "HelloWorld"
```

- Buscar en un archivo:

```
Select-String -Path "archivo.txt" -Pattern "error"
```

4.3. Conceptos Básicos de Expresiones Regulares en PowerShell

4.3.1. Metacaracteres

- .: Coincide con cualquier carácter excepto un salto de línea.
- ^: Coincide con el inicio de una cadena.
- $: Coincide con el final de una cadena.
- *: Coincide con cero o más repeticiones del carácter anterior.
- +: Coincide con una o más repeticiones del carácter anterior.
- ?: Coincide con cero o una repetición del carácter anterior.
- {n,m}: Coincide con al menos n y no más de m repeticiones del carácter anterior.
- []: Define un conjunto de caracteres. Por ejemplo, [a-z] coincide con cualquier letra minúscula.
- () : Agrupa una parte de una expresión regular.

4.3.2. Secuencias de Escape

- \d: Coincide con cualquier dígito.
- \w: Coincide con cualquier carácter de palabra (letra, dígito o guion bajo).
- \s: Coincide con cualquier espacio en blanco (espacio, tabulación, nueva línea).

4.3.3. Grupos y Rangos

- [abc]: Coincide con cualquiera de los caracteres a, b, o c.
- [a-z]: Coincide con cualquier letra minúscula de la a a la z.
- [0-9]: Coincide con cualquier dígito del 0 al 9.

Ejemplos Simples:

- Coincidir con cualquier carácter:

```
"abc" -match "."
```

- Coincidir con el inicio de una cadena:

```
"abc" -match "^a"
```

- Coincidir con un dígito:

```
"123" -match "\d"
```

4.4. Ejemplos Prácticos de Uso de Expresiones Regulares en PowerShell

4.4.1. Validación de Entradas

PowerShell se puede utilizar para validar entradas de usuario mediante expresiones regulares.

Ejemplo: Validar una dirección de correo electrónico:

```
$email = Read-Host "Introduce tu correo electrónico"
if ($email -match '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,}$') {
    Write-Output "Correo válido"
} else {
    Write-Output "Correo no válido"
}
```

4.4.2. Búsqueda y Manipulación de Texto

Utilizando cmdlets como Select-String y operadores como -match, -replace, PowerShell puede buscar y manipular texto de manera efectiva.

Ejemplo: Buscar y resaltar una cadena en un archivo:

```
Select-String -Path "archivo.txt" -Pattern "error"
```

Ejemplo: Extraer direcciones IP de un archivo de log:

```
Select-String -Path "archivo.txt" -Pattern '\b\d{1,3}(\.\d{1,3}){3}\b'
```

Ejemplo: Sustituir direcciones IP en un archivo de log:

```
(Get-Content "archivo.txt") -replace
'\b\d{1,3}(\.\d{1,3}){3}\b', '[IP REDACTADA]' | Set-Content
"archivo.txt"
```

Ejemplo: Obtener estadísticas de uso de recursos en un archivo de log:

```
$logs = Get-Content "archivo.txt"
$cpu_sum = 0
$cpu_count = 0
foreach ($line in $logs) {
    if ($line -match "CPU (\d+)%") {
        $cpu_sum += [int]$matches[1]
        $cpu_count++
    }
}
Write-Output "Promedio de uso de CPU: " + ($cpu_sum / $cpu_count)
```

4.5. Consejos y Buenas Prácticas

- **Prueba tus expresiones regulares:** Antes de utilizarlas en scripts o comandos críticos, prueba tus expresiones regulares en ejemplos pequeños para asegurarte de que funcionan correctamente.
- **Utiliza cmdlets específicos:** PowerShell tiene cmdlets como Select-String, -match, y -replace que están optimizados para trabajar con expresiones regulares.
- **Documenta tus scripts:** Las expresiones regulares pueden ser difíciles de leer y entender. Añade comentarios y documentación a tus scripts para facilitar su mantenimiento.
- **Aprovecha el soporte de .NET:** PowerShell, al estar basado en .NET, permite utilizar la clase [regex] para operaciones avanzadas con expresiones regulares.

4.6. Recursos Adicionales

- **Documentación Oficial:**
  - Documentación de PowerShell
  - Clase .NET Regex
- **Tutoriales en Línea:**
  - PowerShell Regex Cheat Sheet
  - Regular Expressions in PowerShell

Se pide:

1. Elabora un documento explicando qué son las expresiones regulares, sus características y para qué se utilizan.
2. Realiza los ejercicios que se muestran a continuación

Ejercicios

Ejercicio 1: Coincidir con una cadena específica

- **Patrón:** hello
- **Texto:** hello world
- **Explicación:** Este patrón busca la cadena exacta "hello". En el texto dado, "hello" está presente, por lo que coincidirá.

Ejercicio 2: Coincidir con cualquier carácter excepto un salto de línea

- **Patrón:** h.llo
- **Texto:** hello h3llo h_llo
- **Explicación:** El punto (.) coincide con cualquier carácter excepto el salto de línea. El patrón coincide con "hello", "h3llo", y "h_llo" porque el punto puede ser cualquier carácter.

Ejercicio 3: Coincidir con el inicio de una cadena

- **Patrón:** ^start
- **Texto:** start the match
- **Explicación:** El símbolo ^ indica el inicio de la cadena. Este patrón coincide con "start" solo si está al comienzo de la cadena.

Ejercicio 4: Coincidir con el final de una cadena

- **Patrón:** end$
- **Texto:** this is the end
- **Explicación:** El símbolo $ indica el final de la cadena. Este patrón coincide con "end" solo si está al final de la cadena.

Ejercicio 5: Coincidir con uno o más dígitos

- **Patrón:** \d+
- **Texto:** My number is 1234
- **Explicación:** \d coincide con cualquier dígito, y + indica una o más repeticiones. Este patrón coincide con "1234" en el texto.

Ejercicio 6: Coincidir con una secuencia de letras mayúsculas

- **Patrón:** [A-Z]+
- **Texto:** This is an EXAMPLE
- **Explicación:** [A-Z] coincide con cualquier letra mayúscula, y + indica una o más repeticiones. Este patrón coincide con "EXAMPLE".

Ejercicio 7: Coincidir con una secuencia de letras minúsculas

- **Patrón:** [a-z]+
- **Texto:** this is an example
- **Explicación:** [a-z] coincide con cualquier letra minúscula, y + indica una o más repeticiones. Este patrón coincide con "this", "is", "an", y "example".

Ejercicio 8: Coincidir con una secuencia de caracteres alfanuméricos

- **Patrón:** \w+
- **Texto:** abc123
- **Explicación:** \w coincide con cualquier carácter de palabra (letra, dígito o guion bajo), y + indica una o más repeticiones. Este patrón coincide con "abc123".

Ejercicio 9: Coincidir con un espacio en blanco

- **Patrón:** \s
- **Texto:** hello world
- **Explicación:** \s coincide con cualquier espacio en blanco (espacio, tabulación, nueva línea). Este patrón coincide con el espacio entre "hello" y "world".

Ejercicio 10: Coincidir con una dirección de correo electrónico simple

- **Patrón:** [a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}
- **Texto:** contacto@example.com
- **Explicación:** Este patrón coincide con una dirección de correo electrónico que comienza con uno o más caracteres de palabra (letras, dígitos, o ciertos caracteres especiales), seguido de un "@", seguido de uno o más caracteres de palabra, un punto y una extensión de dominio de al menos dos letras. Coincide con "contacto@example.com".

Ejercicio 11: Coincidir con una palabra que comience con 's' y termine con 'e'

- **Patrón:** \bs\w*e\b
- **Texto:** site simple state
- **Explicación:** \b denota un límite de palabra, s indica que la palabra comienza con 's', \w* indica cero o más caracteres de palabra, y e\b indica que la palabra termina con 'e'. Este patrón coincide con "site", "simple", y "state".

Ejercicio 12: Coincidir con números de teléfono en formato (123) 456-7890

- **Patrón:** \(\d{3}\) \d{3}-\d{4}
- **Texto:** (123) 456-7890
- **Explicación:** \(\d{3}\) coincide con tres dígitos entre paréntesis, \d{3} coincide con tres dígitos, y \d{4} coincide con cuatro dígitos. Este patrón coincide con "(123) 456-7890".

Ejercicio 13: Coincidir con una fecha en formato MM/DD/YYYY

- **Patrón:** \b\d{2}/\d{2}/\d{4}\b
- **Texto:** Today's date is 07/18/2024
- **Explicación:** \b denota un límite de palabra, \d{2} coincide con dos dígitos, y \d{4} coincide con cuatro dígitos. Este patrón coincide con "07/18/2024".

Ejercicio 14: Coincidir con una dirección IP

- **Patrón:** \b\d{1,3}(\.\d{1,3}){3}\b
- **Texto:** Server IP is 192.168.1.1
- **Explicación:** \b denota un límite de palabra, \d{1,3} coincide con uno a tres dígitos, y (\.\d{1,3}){3} indica que esto se repite tres veces, separado por puntos. Este patrón coincide con "192.168.1.1".

Ejercicio 15: Coincidir con etiquetas HTML

- **Patrón:** <[^>]+>
- **Texto:** Content
- **Explicación:** < y > coinciden con los caracteres de apertura y cierre de una etiqueta HTML, y [^>]+ indica cualquier carácter excepto > una o más veces. Este patrón coincide con " " y " ".

Ejercicio 16: Coincidir con palabras que contengan 'cat'

- **Patrón:** \b\w*cat\w*\b
- **Texto:** category concatenate
- **Explicación:** \b denota un límite de palabra, \w* indica cero o más caracteres de palabra, cat es el texto que estamos buscando, y \w*\b indica cero o más caracteres de palabra seguidos de un límite de palabra. Este patrón coincide con "category" y "concatenate".

Ejercicio 17: Coincidir con una URL

- **Patrón:** https?://[^\s/$.?#].[^\s]*
- **Texto:** Visit https://www.example.com for more info
- **Explicación:** https? coincide con "http" o "https", :// coincide con "://", [^\s/$.?#] coincide con cualquier carácter que no sea un espacio, barra, dólar, punto, interrogación o almohadilla, y .[^\s]* coincide con un punto seguido de cualquier carácter que no sea un espacio cero o más veces. Este patrón coincide con "https://www.example.com".

Ejercicio 18: Coincidir con líneas que no contienen una palabra específica

- **Patrón:** ^(?!.*forbidden).*$
- **Texto:** This line is allowed
- **Explicación:** ^ indica el inicio de la línea, (? es una negativa lookahead que asegura que el patrón no contiene "forbidden", .* coincide con cualquier carácter cero o más veces, y $ indica el final de la línea. Este patrón coincide con cualquier línea que no contenga la palabra "forbidden".

Ejercicio 19: Coincidir con un código postal de 5 o 9 dígitos (con o sin guión)

- **Patrón:** \b\d{5}(-\d{4})?\b
- **Texto:** ZIP codes: 12345 and 12345-6789
- **Explicación:** \b denota un límite de palabra, \d{5} coincide con cinco dígitos, (-\d{4})? indica un guión seguido de cuatro dígitos opcional, y \b denota un límite de palabra. Este patrón coincide con "12345" y "12345-6789".

Ejercicio 20: Coincidir con una variable en un archivo de configuración (nombre=valor)

- **Patrón:** \b\w+=\w+\b
- **Texto:** user=admin
- **Explicación:** \b denota un límite de palabra, \w+ coincide con uno o más caracteres de palabra, = coincide con el carácter igual, y \w+\b coincide con uno o más caracteres de palabra seguido de un límite de palabra. Este patrón coincide con "user=admin"