Trabajar con expresiones regulares en Python

1. Basic matching with grep

En Python, puedes realizar una búsqueda básica de patrones usando el módulo re. Aquí mostramos cómo verificar si una cadena contiene un patrón específico.

```
import re

# Buscar un patrón en una cadena
pattern = 'hello'
text = 'hello world'
match = re.search(pattern, text)

if match:
    print('Pattern found:', match.group())
else:
    print('Pattern not found')
```

2. Simple matching in Python

Este script muestra cómo hacer una búsqueda simple utilizando expresiones regulares en Python.

```
import re

# Buscar una palabra específica en una cadena
pattern = r'\bworld\b'
text = 'hello world'
match = re.search(pattern, text)

if match:
    print('Match found:', match.group())
```

```
else:
    print('Match not found')
```

3. Wildcards and character classes

Este script utiliza comodines y clases de caracteres para buscar patrones más complejos.

```
import re

# Buscar patrones con comodines y clases de caracteres
pattern = r'gr[aeiou]p'
text = 'grep grip grape'
matches = re.findall(pattern, text)

print('Matches found:', matches)
```

4. Repetition qualifiers

Este script muestra cómo usar los calificadores de repetición para encontrar patrones que se repiten en una cadena.

```
import re

# Buscar patrones con calificadores de repetición
pattern = r'\d{2,4}'
text = 'My numbers are 12, 123, and 1234.'
matches = re.findall(pattern, text)

print('Matches found:', matches)
```

5. Escaping characters

Este script demuestra cómo escapar caracteres especiales en expresiones regulares.

```
import re
```

```
# Buscar un patrón con caracteres especiales
pattern = r'\d+\.\d+'
text = 'The price is 12.34 dollars.'
match = re.search(pattern, text)

if match:
    print('Match found:', match.group())
else:
    print('Match not found')
```

6. Regular expressions in action

Este script muestra cómo aplicar una expresión regular para validar correos electrónicos.

```
import re

# Validar correos electrónicos
pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
email = 'example@example.com'
match = re.match(pattern, email)

if match:
    print('Valid email:', match.group())
else:
    print('Invalid email')
```

7. Capturing groups

Este script utiliza grupos de captura para extraer partes específicas de una cadena.

```
import re

# Utilizar grupos de captura para extraer datos
pattern = r'(\d{2})/(\d{2})/(\d{4})'
text = 'Today's date is 15/08/2023.'
match = re.search(pattern, text)
```

```
if match:
    day, month, year = match.groups()
    print(f'Day: {day}, Month: {month}, Year: {year}')
else:
    print('No match found')
```

8. More on repetition qualifiers

Este script muestra cómo usar diferentes calificaciones de repetición en expresiones regulares.

```
import re

# Buscar patrones con diferentes calificaciones de repetición
pattern = r'\b\w+\b'
text = 'This is a test.'
matches = re.findall(pattern, text)

print('Words found:', matches)
```

9. Extracting a PID using regexes in Python

Este script muestra cómo extraer un PID (Process ID) de una cadena de texto utilizando expresiones regulares.

```
import re

# Patrón para encontrar números en el texto
pattern = r'\d+'
text = 'The current PID is 12345.'

matches = re.findall(pattern, text)
print('All numbers found:', matches)
```