PYTHON

Expresiones regulares

EXPRESIONES REGULARES

- Módulo **re**
  - Pueden operar con cadenas de caracteres (str) o con bytes.
  - Normalmente los patrones se expresan utilizando *raw strings.*
  - Las expresiones utilizan la barra inversa '\' para utilizar caracteres especiales ignorando su significado.
  - La expresiones regulares se suelen utilizar a través de expresiones regulares compiladas (mediante funciones y métodos a nivel de módulo).
  - Caracteres especiales:
    - https://docs.python.org/3/library/re.html (inglés)
    - https://docs.python.org/es/3/library/re.html (español)

EXPRESIONES REGULARES

- Módulo **re**
  - Uso de *raw string*:
    - Evita los significados de los caracteres especiales en Python, simplificando la construcción de patrones.

```
import re
regex = (r"[a-z]+\\[a-z]+")
cadena = ("casa\piso")
correcta = re.fullmatch(regex, cadena)
print(correcta)
```

EXPRESIONES REGULARES

- Módulo **re**
  - Objeto re.Match:
    - Método span: contiene una tupla con las posiciones de inicio y fin de la subcadena que cumple el patrón.
    - Método group: contiene la subcadena que cumple el patrón.
    - Métodos start y end. Posiciones iniciales y finales de la ocurrencia.

```
import re
exp = "[a-z]+"
re_object = re.compile(exp)
ocurrencias = re_object.finditer("ax3b8c")
for ocurrencia in ocurrencias:
    print(ocurrencia.span())#Posiciones
    print(ocurrencia.group())#Contenido
```

```
(0, 2)
ax
(3, 4)
b
(5, 6)
c
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - compile→compila un patrón de una expresión regular y genera un objeto de expresión regular (re.Pattern). Este objeto se puede utilizar en las funciones **match** y **search**

```
import re
exp = "[a-z]"
re_object = re.compile(exp)
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - search→busca el primer patrón de la expresión en una cadena. Retorna un objeto match (re.Match) o None.

```
retorno = re_object.search("3abc")
```

    - match→busca el patrón de la expresión en el inicio de la cadena. Retorna un objeto match o None.

```
retorno = re_object.match("3abc")
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - Las funciones se pueden ejecutar sobre expresiones compiladas o sin compilar.

```
import re
exp = "[a-z]+"
re_object = re.compile(exp)
#Sobre la expresión compilada
resultado = re_object.search("38ab42*")
print(resultado)
#Proporcionando la expresión
resultado = re.search("[a-z]+", "38ab42*")
print(resultado)
```

```
<re.Match object; span=(2, 4), match='ab'>
<re.Match object; span=(2, 4), match='ab'>
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - Las funciones admiten flags para modificar el comportamiento por defecto.

```
import re
resultado = re.search("[a-z]+", "38AB42*")
print(resultado)
```

```
None
```

```
import re
resultado = re.search("[a-z]+", "38AB42*", re.IGNORECASE)
print(resultado)
```

```
<re.Match object; span=(2, 4), match='AB'>
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - Las funciones admiten flags para modificar el comportamiento por defecto.
    - MULTILINE: aplica los operadores ^ (comienzo) y $ (fin) a cada línea en lugar de a la cadena completa.

```
import re
entrada = """125 PALABRA
258 OTRA
"""
expresion = r"(^[0-9]+)"

ocurrencias = re.finditer(expresion, entrada)
for ocurrencia in ocurrencias:
    print("Por defecto:", ocurrencia)

ocurrencias = re.finditer(expresion, entrada, re.MULTILINE)
for ocurrencia in ocurrencias:
    print("Multilínea:", ocurrencia)
```

Por defecto: <re.Match object; span=(0, 3), match='125'>

Multilínea: <re.Match object; span=(0, 3), match='125'>
Multilínea: <re.Match object; span=(12, 15), match='258'>

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - fullmatch→busca el patrón de la expresión en la cadena completa. Retorna un objeto match o None.

```
retorno = re_object.fullmatch("3abc")
```

    - split→divide la cadena utilizando el patrón como separador.

```
import re
exp = "F"
re_object = re.compile(exp)
retorno = re_object.split("aFbFc")
print(retorno)

['a', 'b', 'c']
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - findall→busca todas las ocurrencias no superpuestas del patrón en una cadena.

```
import re
exp = "[a-z]"
re_object = re.compile(exp)
retorno = re_object.findall("a3b8c")
print(retorno)

['a', 'b', 'c']
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - finditer→Retorna un **iterator** con las coincidencias no superpuestas del patrón encontradas en una cadena.

```
import re
exp = "[a-z]+"
re_object = re.compile(exp)
ocurrencias = re_object.finditer("ax3b8c")
for ocurrencia in ocurrencias:
    pass
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - sub→Reemplaza las ocurrencias encontradas en una cadena por la cadena indicada. Devuelve un string.

```
import re
exp = "[a-z]+"
re_object = re.compile(exp)
resultado = re_object.sub("?", "ax3b8c")
print(resultado)
```

```
?3?8?
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - subn→Reemplaza las ocurrencias encontradas en una cadena por la cadena indicada. Devuelve una tupla con el resultado y el número de ocurrencias encontradas.

```
import re
exp = "[a-z]+"
re_object = re.compile(exp)
resultado = re_object.subn("?", "ax3b8c")
print(resultado)
```

```
('?3?8?', 3)
```

EXPRESIONES REGULARES

- Módulo **re**
  - Funciones:
    - escape→Sustituye los caracteres con significado por su equivalente 'escapado'.

```
resultado = re.escape('https://www.python.org')
print(resultado)
```

```
https://www\.python\.org
```

    - purge→Limpia la caché de la expresión regular.

EXPRESIONES REGULARES

- Módulo **re.**
  - Ejercicios y ejemplos.

EXPRESIONES REGULARES

- Enlaces:
  - https://docs.python.org/3/library/re.html
  - https://docs.python.org/3/howto/regex.html#regex-howto
  - https://developers.google.com/edu/python/regular-expressions
  - https://www.w3schools.com/python/python_regex.asp
  - https://www.tutorialspoint.com/python/python_reg_expressions.htm
  - https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/
  - https://www.geeksforgeeks.org/regular-expressions-python-set-1-search-match-find/?ref=lbp
  - https://www.geeksforgeeks.org/python-regex-re-search-vs-re-findall/?ref=lbp
  - https://realpython.com/regex-python/
  - https://realpython.com/regex-python-part-2/