Regex en Python

Ou como utilizar regexes sen medo

M. Torre Castro[1]

[1]Python Coruña

Charlas Python Coruña, 2022/23

Índice

Outline

Entradas incorrectas do usuario

Os usuarios son humanos e cometen erros. E a veces son demasiado humanos...

- Moitas veces a información que se introduce desde fora do noso software acaba na BD
- Calquer entrada dun usuario pode ter un erro
- Hai información que debe cumprir un formato consistente pra poder manexala

Entradas incorrectas do usuario

Os usuarios son humanos e cometen erros. E a veces son demasiado humanos...

- Moitas veces a información que se introduce desde fora do noso software acaba na BD
- Calquer entrada dun usuario pode ter un erro
- Hai información que debe cumprir un formato consistente pra poder manexala

Entradas incorrectas do usuario

Os usuarios son humanos e cometen erros. E a veces son demasiado humanos...

- Moitas veces a información que se introduce desde fora do noso software acaba na BD
- Calquer entrada dun usuario pode ter un erro
- Hai información que debe cumprir un formato consistente pra poder manexala

Un exemplo: Unha data

- Nos interesan datas introducidas polo usuario
- Pero seguindo un formato, imos poñer o seguinte: dd/MM/yyyy
- Cómo resolveríamos esta verificación en Python?

Un exemplo: Unha data

- Nos interesan datas introducidas polo usuario
- Pero seguindo un formato, imos poñer o seguinte: dd/MM/yyyy
- Cómo resolveríamos esta verificación en Python?

Un exemplo: Unha data

- Nos interesan datas introducidas polo usuario
- Pero seguindo un formato, imos poñer o seguinte: dd/MM/yyyy
- Cómo resolveríamos esta verificación en Python?

Proposta inicial

```
input_date = input('Introduza unha data, por favor: ')

"""
Primer problema, como dividir o string en tres:
    dia, mes e ano
separar polas barras '/' parece lóxico
pero, ¿e se houbo un erro de tecleado?
"""
dia, mes, ano = input_date.split('/')
```

Proposta inicial

```
# Agora pra cada parte, debemos verificar a validez
# e que o valor está nun rango

try:
    if int(ano) <= getCurrentYear():
        ano = int(ano)

    if 0 < int(mes) < 13:
        mes = int(mes)

    if 0 < int(dia) < getDaysOfMonth(mes):
        dia = int(dia)
except ValueError:
    # Xestion do erro
```

Solución con regex

```
import re

DATE_REGEX = r'(?P<dia>0[1-9]|1[0-9]|2[0-9]|3[01])/'
DATE_REGEX += r'(?P<mes>0[1-9]|1[012])/'
DATE_REGEX += r'(?P<ano>[0-9]{4})'
DATE_REGEX_PATTERN = re.compile(DATE_REGEX)

input_date = input('Introduza unha data, por favor: ')
match = DATE_REGEX_PATTERN.match(input_date)

if match:
    # Xa sabemos que toda a data é correcta
    # Só queda comprobar os días por mes
    mes = match.group('mes')
    dia = match.group('dia')
    if dia > getDaysOfMonth(mes):
        # ...codigo avisando do erro...
```

Vantaxes

- Reducción de código
- Comprobación completa do formato da info de entrada nunha soa operación
- Extracción das tres partes do dato na mesma operación

Vantaxes

- Reducción de código
- Comprobación completa do formato da info de entrada nunha soa operación
- Extracción das tres partes do dato na mesma operación

Vantaxes

- Reducción de código
- Comprobación completa do formato da info de entrada nunha soa operación
- Extracción das tres partes do dato na mesma operación

Desvantaxes

Figure:

Outline

Qué é un regex?

Expresión regular É un tipo de expresión con sintaxe propia que contén un patrón de búsqueda pra un texto

De agora en adiante imos explicar con exemplos as expresións regulares. Normalmente marcaremos con comillas simples as regexes e en verde a parte da cadena que coincide.

Sentidevos libres de preguntar en calquer momento ou suxerir calquer variante.

Sintaxe

- Un carácter. Exemplo: r'a' fai coincidencia con 'Manuel Arias'
- Pode facer match mais dunha vez se llo dicimos ao motor de regex
- Varios caracteres. Ex: r'afei' fai coincidencia con 'afeitar' pero non con 'afectar'.
- r'^' e r'$' indican principio e final de liña.
- Ex: r'^python$' fai match con 'python', pero non con '␣python' ou 'python␣'

Sintaxe

- Un carácter. Exemplo: r'a' fai coincidencia con 'Manuel Arias'
- Pode facer match mais dunha vez se llo dicimos ao motor de regex
- Varios caracteres. Ex: r'afei' fai coincidencia con 'afeitar' pero non con 'afectar'.
- r'^' e r'$' indican principio e final de liña.
- Ex: r'^python$' fai match con 'python', pero non con '␣python' ou 'python␣'

Sintaxe

- Un carácter. Exemplo: r'a' fai coincidencia con 'Manuel Arias'
- Pode facer match mais dunha vez se llo dicimos ao motor de regex
- Varios caracteres. Ex: r'afei' fai coincidencia con 'afeitar' pero non con 'afectar'.
- r'^' e r'$' indican principio e final de liña.
- Ex: r'^python$' fai match con 'python', pero non con '␣python' ou 'python␣'

Sintaxe

- '[]' indican alternancia. Ex: r'[ei]u' fai coincidencia con 'correu' e 'saiu'
- Podense definir rangos dentro. Ex: r'[a-z]-[0-9]' fai match 'k-2'
- [^a] busca o que non coincida con 'a'. Ex: r'[^a-zA-Z]' coincide con 'Python3'

Sintaxe

- '[]' indican alternancia. Ex: r'[ei]u' fai coincidencia con 'correu' e 'saiu'
- Podense definir rangos dentro. Ex: r'[a-z]-[0-9]' fai match 'k-2'
- [^a] busca o que non coincida con 'a'. Ex: r'[^a-zA-Z]' coincide con 'Python3'

Sintaxe

- \d coincide con díxito
- \w con caracter de palabra
- \s con caracter “invisible”

Sintaxe

- \d coincide con díxito
- \w con caracter de palabra
- \s con caracter “invisible”

Sintaxe

- \d coincide con díxito
- \w con caracter de palabra
- \s con caracter “invisible”

Sintaxe

- Ex: r'\d\sgoles' fai coincidencia con 'O xogador marcou 3␣goles'
- '.' encaixa con case calquer caracter. Ex: r'.atata' fai match con 'patata' e 'batata'

Sintaxe

- Ex: r'\d\sgoles' fai coincidencia con 'O xogador marcou 3␣goles'
- '.' encaixa con case calquer caracter. Ex: r'.atata' fai match con 'patata' e 'batata'

Sintaxe

- ? indica opcionalidade. Ex: r'H?[eE]lena' fai match con 'Helena' e 'Elena'
- * indica repetición **optativa**. Ex: r'Tel: \d*' fai match con 'Tel: 981987654' pero tamén con 'Tel: '
- + indica repetición **necesaria**. Ex: r'Tel: \d+' fai match con 'Tel: 981987654' pero non con 'Tel: '

Sintaxe

- ? indica opcionalidade. Ex: r'H?[eE]lena' fai match con 'Helena' e 'Elena'
- * indica repetición **optativa**. Ex: r'Tel: \d*' fai match con 'Tel: 981987654' pero tamén con 'Tel: '
- + indica repetición **necesaria**. Ex: r'Tel: \d+' fai match con 'Tel: 981987654' pero non con 'Tel: '

Sintaxe

- ? indica opcionalidade. Ex: r'H?[eE]lena' fai match con 'Helena' e 'Elena'
- * indica repetición **optativa**. Ex: r'Tel: \d*' fai match con 'Tel: 981987654' pero tamén con 'Tel: '
- + indica repetición **necesaria**. Ex: r'Tel: \d+' fai match con 'Tel: 981987654' pero non con 'Tel: '

Sintaxe

- A repetición pode ser *"voraz"* ou non. '+' e '*' poden combinarse con '?' pra que non sexa voraz.
- Ex: r'<.+>' fai match con '<title>Un título</title>'
- Ex: r'<.+?>' fai match con '<title>Un título</title>'

Sintaxe

- A repetición pode ser *"voraz"* ou non. '+' e '*' poden combinarse con '?' pra que non sexa voraz.
- Ex: r'<.+>' fai match con '<title>Un título</title>'
- Ex: r'<.+?>' fai match con '<title>Un título</title>'

Sintaxe

- A repetición pode ser *"voraz"* ou non. '+' e '*' poden combinarse con '?' pra que non sexa voraz.
- Ex: r'<.+>' fai match con '<title>Un título</title>'
- Ex: r'<.+**?**>' fai match con '<title>Un título</title>'

Sintaxe

- Por último, os parénteses crean “grupos” e capturan o seu contido.
- Ex: Pra un DNI, r’(\d+)-(.)’ captura o número e letra. ’32840832-G’
- O número ira no grupo “1” e a letra no grupo “2”
- Hai moitos moitos tipos de grupos que podedes consultar nas docs

Sintaxe

- Por último, os parénteses crean "grupos" e capturan o seu contido.
- Ex: Pra un DNI, r'(\d+)-(.)' captura o número e letra. '32840832-G'
- O número ira no grupo "1" e a letra no grupo "2"
- Hai moitos moitos tipos de grupos que podedes consultar nas docs

Sintaxe

- Por último, os parénteses crean "grupos" e capturan o seu contido.
- Ex: Pra un DNI, r'(\d+)-(.)' captura o número e letra. '32840832-G'
- O número ira no grupo "1" e a letra no grupo "2"
- Hai moitos moitos tipos de grupos que podedes consultar nas docs

Sintaxe

- Por último, os parénteses crean "grupos" e capturan o seu contido.
- Ex: Pra un DNI, r'(\d+)-(.)' captura o número e letra. '32840832-G'
- O número ira no grupo "1" e a letra no grupo "2"
- Hai moitos moitos tipos de grupos que podedes consultar nas docs

Outline

O módulo re

Figure:

O módulo re

- O modulo re é o encargado en Python de facer pattern-matching con regexes
- Provee métodos pra o anterior. Os máis importantes:

  compile() Crea un obxeto de regex pra facer comparacións contra cadenas/strings

  match() Compara **desde o inicio** da cadena. Ex: 'facer' fai match con 'facer' pero **non con** 'desfacer'

  search() Busca dentro da cadena. Ex: 'brazo' fai match con 'abrazos'

  findall() Como search(), pero segue buscando trala primeira coincidencia e devolve unha lista

  group() Devolve un grupo de coincidencia

O módulo re

- O modulo re é o encargado en Python de facer pattern-matching con regexes
- Provee métodos pra o anterior. Os máis importantes:

  compile() Crea un obxeto de regex pra facer comparacións contra cadenas/strings

  match() Compara **desde o inicio** da cadena. Ex: 'facer' fai match con 'facer' pero **non con** 'desfacer'

  search() Busca dentro da cadena. Ex: 'brazo' fai match con 'abrazos'

  findall() Como search(), pero segue buscando trala primeira coincidencia e devolve unha lista

  group() Devolve un grupo de coincidencia

O módulo re

- O modulo re é o encargado en Python de facer pattern-matching con regexes
- Provee métodos pra o anterior. Os máis importantes:

compile()	Crea un obxeto de regex pra facer comparacións contra cadenas/strings
match()	Compara **desde o inicio** da cadena. Ex: 'facer' fai match con 'facer' pero **non con** 'desfacer'
search()	Busca dentro da cadena. Ex: 'brazo' fai match con 'abrazos'
findall()	Como search(), pero segue buscando trala primeira coincidencia e devolve unha lista
group()	Devolve un grupo de coincidencia

O módulo re

- O modulo re é o encargado en Python de facer pattern-matching con regexes
- Provee métodos pra o anterior. Os máis importantes:

compile()	Crea un obxeto de regex pra facer comparacións contra cadenas/strings
match()	Compara **desde o inicio** da cadena. Ex: 'facer' fai match con 'facer' pero **non con** 'desfacer'
search()	Busca dentro da cadena. Ex: 'brazo' fai match con 'abrazos'
findall()	Como search(), pero segue buscando trala primeira coincidencia e devolve unha lista
group()	Devolve un grupo de coincidencia

O módulo re

- O modulo re é o encargado en Python de facer pattern-matching con regexes
- Provee métodos pra o anterior. Os máis importantes:

compile()	Crea un obxeto de regex pra facer comparacións contra cadenas/strings
match()	Compara **desde o inicio** da cadena. Ex: 'facer' fai match con 'facer' pero **non con** 'desfacer'
search()	Busca dentro da cadena. Ex: 'brazo' fai match con 'abrazos'
findall()	Como search(), pero segue buscando trala primeira coincidencia e devolve unha lista
group()	Devolve un grupo de coincidencia

O módulo re

- O modulo re é o encargado en Python de facer pattern-matching con regexes
- Provee métodos pra o anterior. Os máis importantes:

compile()	Crea un obxeto de regex pra facer comparacións contra cadenas/strings
match()	Compara **desde o inicio** da cadena. Ex: 'facer' fai match con 'facer' pero **non con** 'desfacer'
search()	Busca dentro da cadena. Ex: 'brazo' fai match con 'abrazos'
findall()	Como search(), pero segue buscando trala primeira coincidencia e devolve unha lista
group()	Devolve un grupo de coincidencia

Escape

- As regexes se definen en strings, pero os díxitos invisibles como \n, \t levan contrabarra
- Os díxitos especiais xa vistos \w, \d ... tamén levan contrabarra
- A solución pra non ter que "poblar" todo con contrabarras e marcar o string de Python como r"
- Así Python sabe que non debe manexar '\' de modo especial e o módulo **re** fai todo

Escape

- As regexes se definen en strings, pero os díxitos invisibles como \n, \t levan contrabarra
- Os díxitos especiais xa vistos \w, \d ... tamén levan contrabarra
- A solución pra non ter que "poblar" todo con contrabarras e marcar o string de Python como r"
- Así Python sabe que non debe manexar '\' de modo especial e o módulo **re** fai todo

Escape

- As regexes se definen en strings, pero os díxitos invisibles como \n, \t levan contrabarra
- Os díxitos especiais xa vistos \w, \d ... tamén levan contrabarra
- A solución pra non ter que "poblar" todo con contrabarras e marcar o string de Python como r"
- Así Python sabe que non debe manexar '\' de modo especial e o módulo **re** fai todo

Escape

- As regexes se definen en strings, pero os díxitos invisibles como \n, \t levan contrabarra
- Os díxitos especiais xa vistos \w, \d ... tamén levan contrabarra
- A solución pra non ter que "poblar" todo con contrabarras e marcar o string de Python como r"
- Así Python sabe que non debe manexar '\' de modo especial e o módulo **re** fai todo

Outline

Escribindo regexes

Figure:

Exemplo 1

```
import re
PATH_REGEX = re.compile(r'c:\\temp')
answer = input('Cal é a ruta temporal en Windows?: ').strip()
match = PATH_REGEX.match(answer)
if match is None:
    print('Resposta incorrecta')
else:
    print('Resposta correcta')
```

Exemplo 2

```
import re
text = 'Tres tristes tigres comian trigo"
TR_REGEX = re.compile(r'[Tt]r\w*')
result_list = TR_REGEX.findall(text)
for element in result_list:
    print(element)
```

Exemplo 3

```
import re
text = 'Tres tristes tigres comian trigo"
TR_REGEX = re.compile(r'[Tt]r\w*')
result_list = TR_REGEX.findall(text)
for element in result_list:
    print(element)
```

O límite está na imaxinación

- DNIs
- Datas, horas
- E-mails
- Teléfonos
- Colores en hexadecimal
- Tarxetas de crédito
- Códigos de barra
- ...e moito máis ...

Non esaxerar, ho!

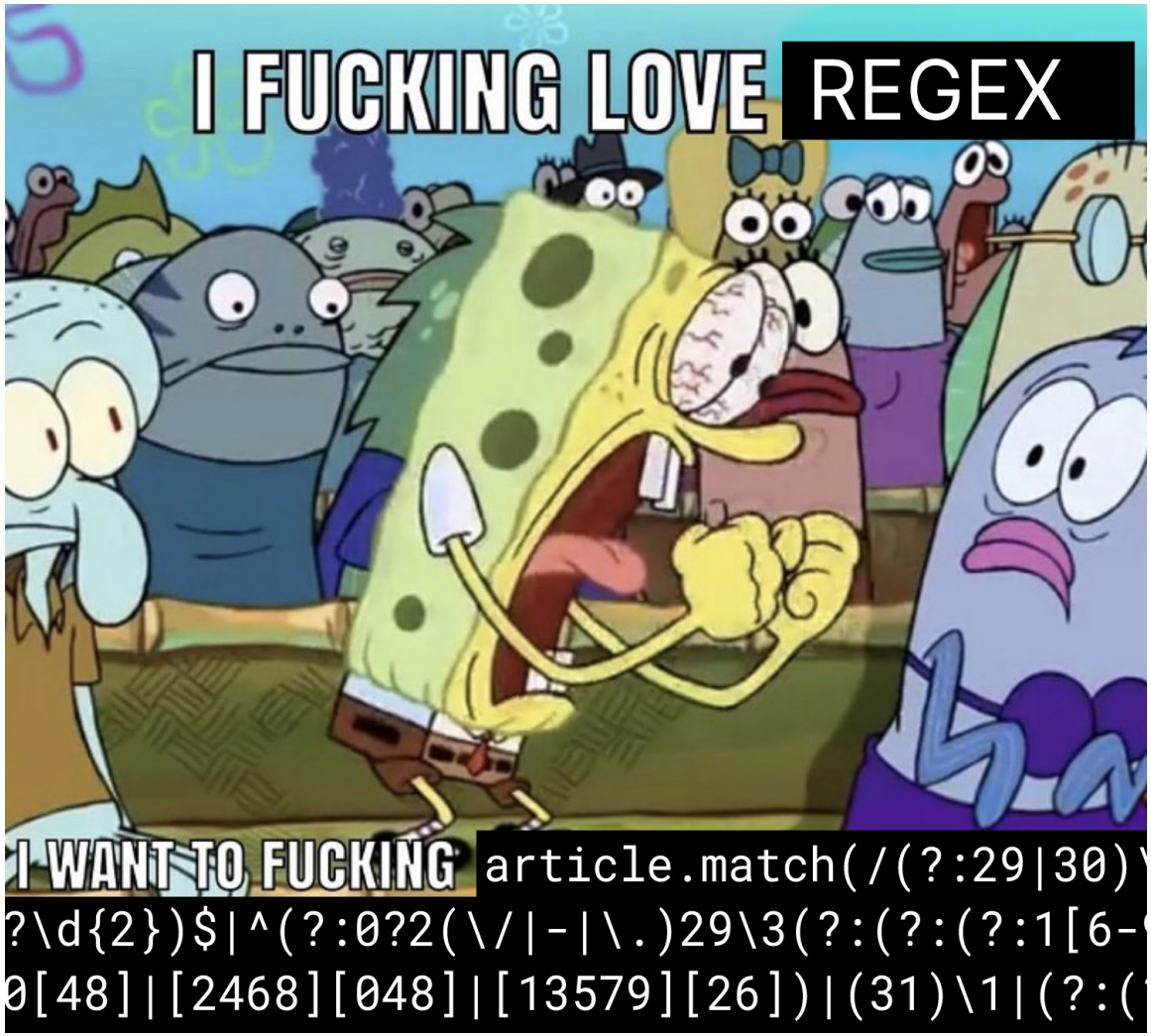

Figure:

Resumo

- Agora coñecedes as bases das expresións regulares. ¿Cándo usalas?
- Operar con cadenas, concatenar, comparar cadenas segue sendo necesario
- O máis recomendable é usar regex cando a info ven nun formato moi concreto
- ...ou cando se busca un contido moi especifico, pero que pode vir con variantes

Resumo

- Agora coñecedes as bases das expresións regulares. ¿Cándo usalas?
- Operar con cadenas, concatenar, comparar cadenas segue sendo necesario
- O máis recomendable é usar regex cando a info ven nun formato moi concreto
- ...ou cando se busca un contido moi especifico, pero que pode vir con variantes

Resumo

- Agora coñecedes as bases das expresións regulares. ¿Cándo usalas?
- Operar con cadenas, concatenar, comparar cadenas segue sendo necesario
- O máis recomendable é usar regex cando a info ven nun formato moi concreto
- ...ou cando se busca un contido moi especifico, pero que pode vir con variantes

Resumo

- Agora coñecedes as bases das expresións regulares. ¿Cándo usalas?
- Operar con cadenas, concatenar, comparar cadenas segue sendo necesario
- O máis recomendable é usar regex cando a info ven nun formato moi concreto
- ...ou cando se busca un contido moi especifico, pero que pode vir con variantes

Pra ler máis I

- https://docs.python.org/3/library/re.html.
- https://docs.python.org/3/howto/regex.html#regex-howto
- *https://www.regular-expressions.info*
- https://www.html5pattern.com/