What is Git?

It is one flavor of version control. In fact, it's what's known as a distributed version control system

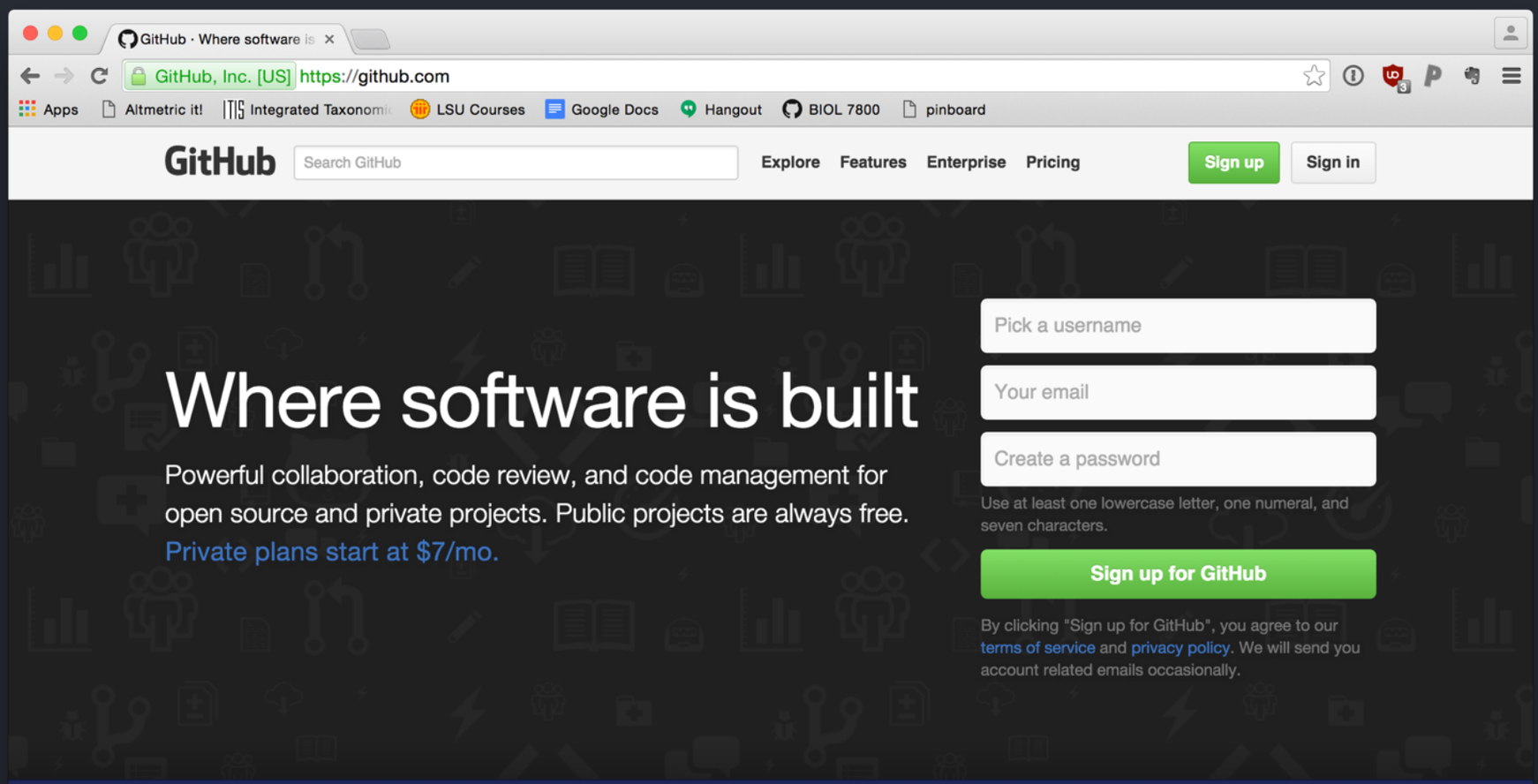
(it doesn't depend on a central store of information)



Created by Linus Torvalds
to help maintain the **Linux**
operating system kernel

What is Github?

It is basically a web “wrapper” around many features of **git** that also include features to help track bugs, collaborate on features, manage tasks, and share code.



But why are we *using* all of this?

Largely because that's how it's done and I want to introduce you to **the tools** you'll run into now, rather than waiting **until later**.

There will also come a time when you are **very, very happy** for the ability to try out new things, step back in time, and use previous commits to identify errors.

Better to start good practices today than in a few years (or never).

Code Review !

How we’re gonna do it…

Check out Alicia’s ipynb

Look at Grace’s directory

Check out May’s answer to range question

Check out Mike’s answer to randint question

Check out Amie’s answer to randint question

Pseudocode & Regular Expressions

Programming (for biologists)

BIOL 7800

Pseudocode

What is it?

basically, *English*

Very, very, very *high-level* description of the **likely steps** that your program (or function) will take to achieve a goal.

Pseudocode

What does it look like?

language-style (less formal)

untitled

untitled

```
1    Sub fizzbuzz()
2      For i = 1 to 100
3        print_number = True
4        If i is divisible by 3 Then
5          Print "Fizz"
6          print_number = False
7        End If
8        If i is divisible by 5 Then
9          Print "Buzz"
10         print_number = False
11       End If
12       If print_number = True Then print i
13       Print a newline
14     Next i
15   End Sub
```

File 0 Project 0 ✔ No Issues untitled* 1:1 LF UTF-8 Python

“mathematical”

untitled

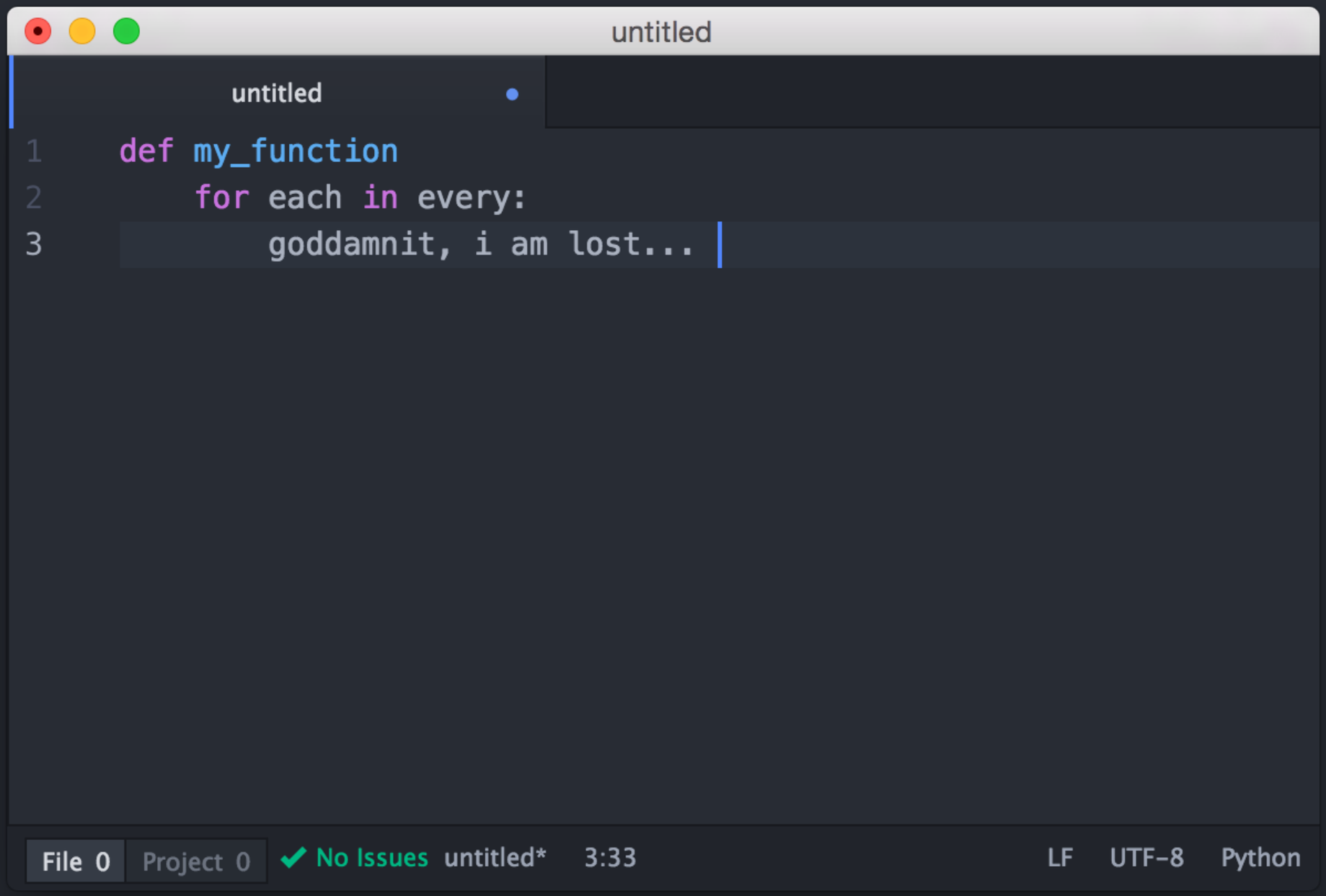
untitled

```
1   algorithm ford-fulkerson is
2       input: Graph G with flow capacity c,
3              source node s,
4              sink node t
5       output: Flow f such that f is maximal from s to t
6
7       (Note that f(u,v) is the flow from node u to node v, and c(u,v) is the flow
8
9       for each edge (u, v) in GE do
10          f(u, v) ← 0
11          f(v, u) ← 0
12
13      while there exists a path p from s to t in the residual network Gf do
14          let cf be the flow capacity of the residual network Gf
15          cf(p) ← min{cf(u, v) | (u, v) in p}
16          for each edge (u, v) in p do
17              f(u, v) ←  f(u, v) + cf(p)
18              f(v, u) ← -f(u, v)
19
20      return f
```

File 0 Project 0 ✔ No Issues untitled* 1:1

Pseudocode

What does it look like?



Pseudocode

What does it look like?

untitled

untitled

```
def my_function
    read in file containing sequence data
    for each_read in my_sequence_data:
        extract the header information
        reformat the header to remove the sequence tag
        write the data out to a file

def my_other_function
    read in file containing sequence data
    for each_read in my_sequence data:
        read in sequence
        convert sequence to reverse complement
        convert any bases with QUAL < 15 to "N"
```

File 0 | Project 0 | ✔ No Issues | untitled* | 13:48 | LF | UTF-8 | Python

Pseudocode

What does it look like?

untitled

untitled

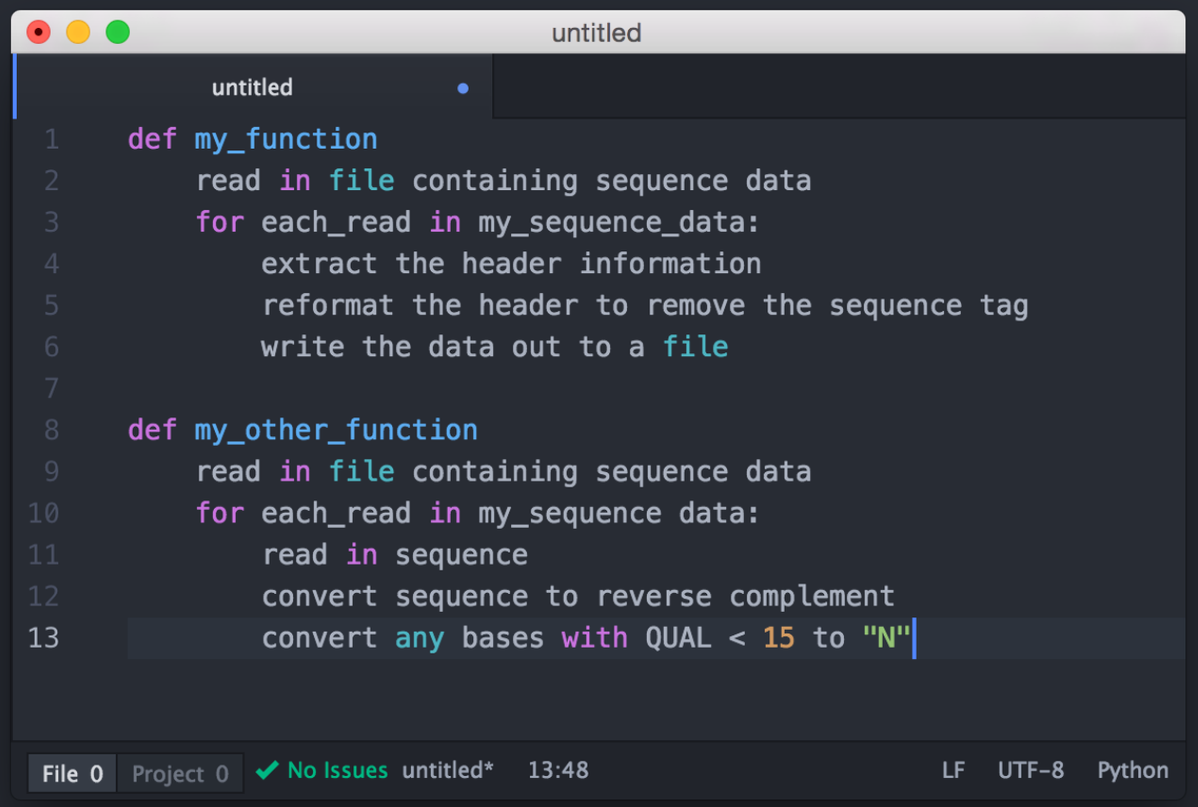
```
1    def my_function
2        #write the data out to a file
3        with open("output.fasta", "w") as outfile:
4        #read in file containing sequence data
5        data = SeqIO.parse(open("myfile.fasta"), 'fasta')
6        #for each_read in my_sequence_data:
7        for sequence in data:
8            #extract the header information
9            header = sequence.description
10           #reformat the header to remove the sequence tag
11           header = header.split(":")[:-1]
12           header = ":".join(header)
13           sequence.description = header
14           outfile.write(sequence.format("fasta"))
15
```

File 0 | Project 0 | ✔ No Issues | untitled* | 9:42 | LF | UTF-8 | Python

Pseudocode

What should I use it for?

It can help you lay out a logical path forward, very similar to outlining a manuscript.



(it can also help you think through a **mental road block**)

Now for something different



Banksy

Regular Expressions

Among the **most hated** programming experiences!

And, possibly, one of the most **useful** things you'll learn.



theoatmeal.com

Regular Expressions

What are they?

```
\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b
```

Matches ***any*** email address ?

source: http://www.regular-expressions.info/email.html

Regular Expressions

What are they?

“a **sequence of characters** that define a **search pattern**, mainly for use in pattern matching with strings, or string matching, i.e. "find and replace"-like operations.”

source: wikipedia

“A regular expression (***regex*** or ***regexp*** for short) is a **special text string** for describing a search pattern.
You can think of regular expressions as **wildcards on steroids**.

source: http://www.regular-expressions.info

Regular Expressions

What are they?

“a **sequence of characters** that define a **search pattern**, mainly for use in pattern matching with strings, or string matching, i.e. "find and replace"-like operations.”

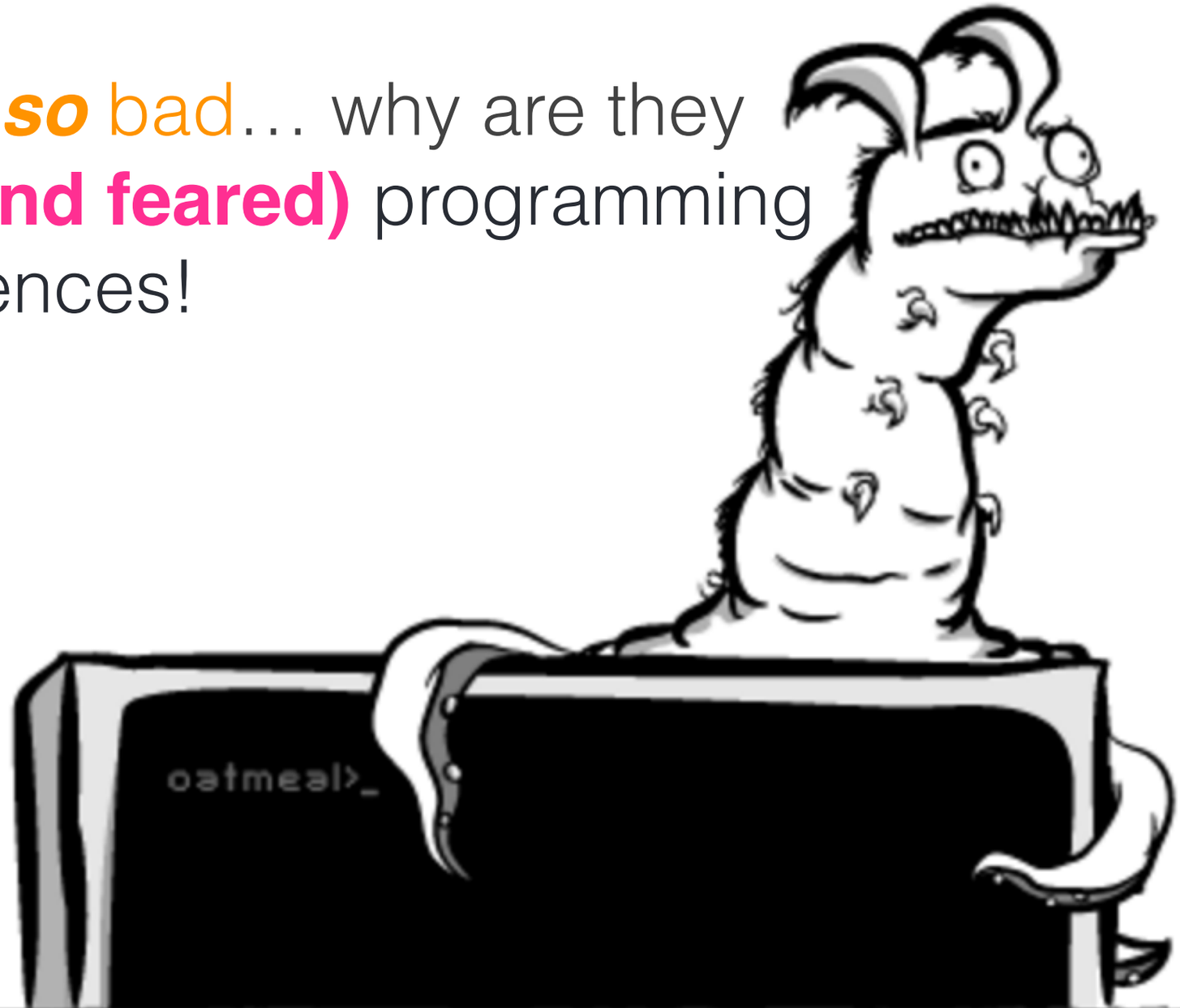
source: wikipedia

“A regular expression (***regex*** or ***regexp*** for short) is a **special text string** for describing a search pattern.
You can think of regular expressions as **wildcards on steroids**.

source: http://www.regular-expressions.info

Regular Expressions

Well, that doesn't seem ***so*** bad… why are they among the **most hated (and feared)** programming experiences!



theoatmeal.com

Regular Expressions

Why do they **suck**?

they're **hard**

they're **complex**

they're both **similar** and **different** (by implementation). **surprise!**

Regular Expressions

Why do they **suck**?

they're **hard**

text is, perhaps not surprisingly, often variable and hard to match

we're applying a discrete classification to a variable problem

Brant **C.** **F**aircloth	**B**rant **c** **F**aircloth	**b**rant **c.** **f**airclot**H**	
225-555-1212	225.555.1212	**1.**225.555.1212	**+01.**225.555-1212

Regular Expressions

Why do they **suck**?

they're **complex**

Sometimes we want to match	**Sometimes we want to capture**	**Sometimes we want to exclude**
georgia bulldogs	georgia bulldogs	georgia bulldogs
(?:georgia)	(bulldogs)	[^g,e,o,l,\s]
georgia bulldogs	georgia **bulldogs**	geo**rgia bu**ll**d**og**s**

Regular Expressions

Why do they **suck**?

they're both **similar** and **different** (by implementation). **surprise!**

Onigurama (Atom)

untitled

untitled

```
1   1. Syntax elements
2
3   \       escape (enable or disable meta character meaning)
4   |       alternation
5   (...)   group
6   [...]   character class
7
8   3. Character types
9
10  .        any character (except newline)
11  \w       word character
12  \W       non word char
13  \s       whitespace char
14  \S       non whitespace char
15  \d       decimal digit char
16  \D       non decimal digit char
17  \h       hexadecimal digit char    [0-9a-fA-F]
```

Python

untitled

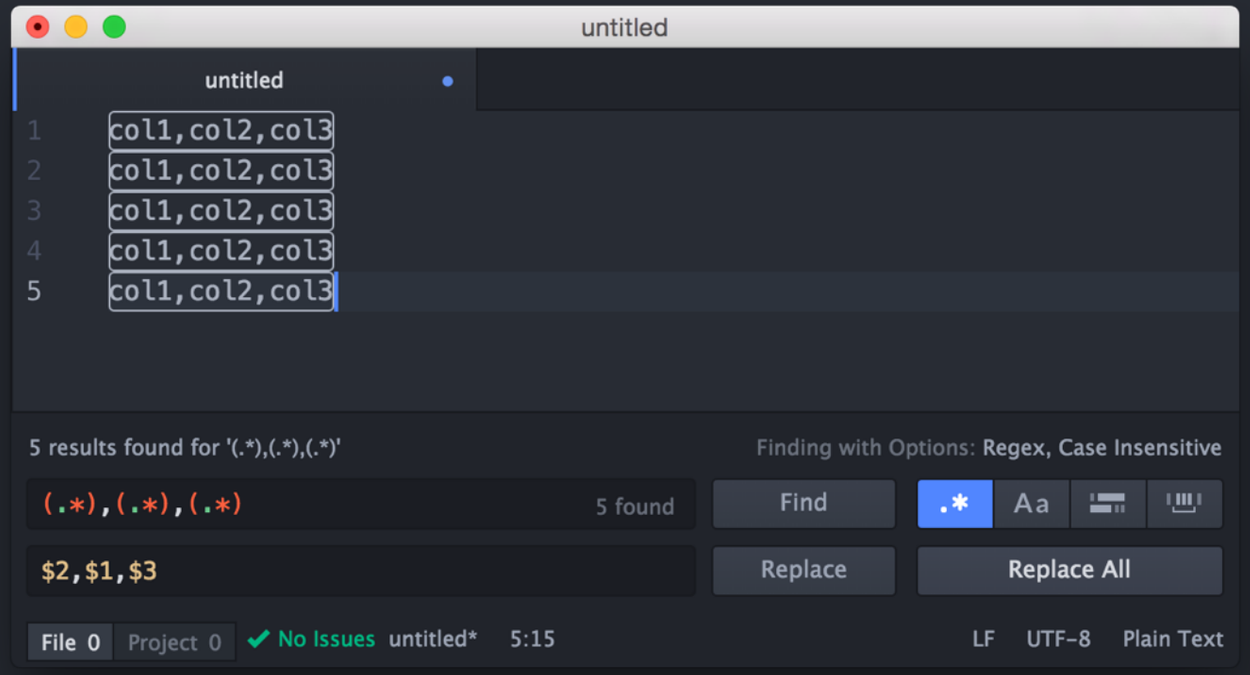
untitled

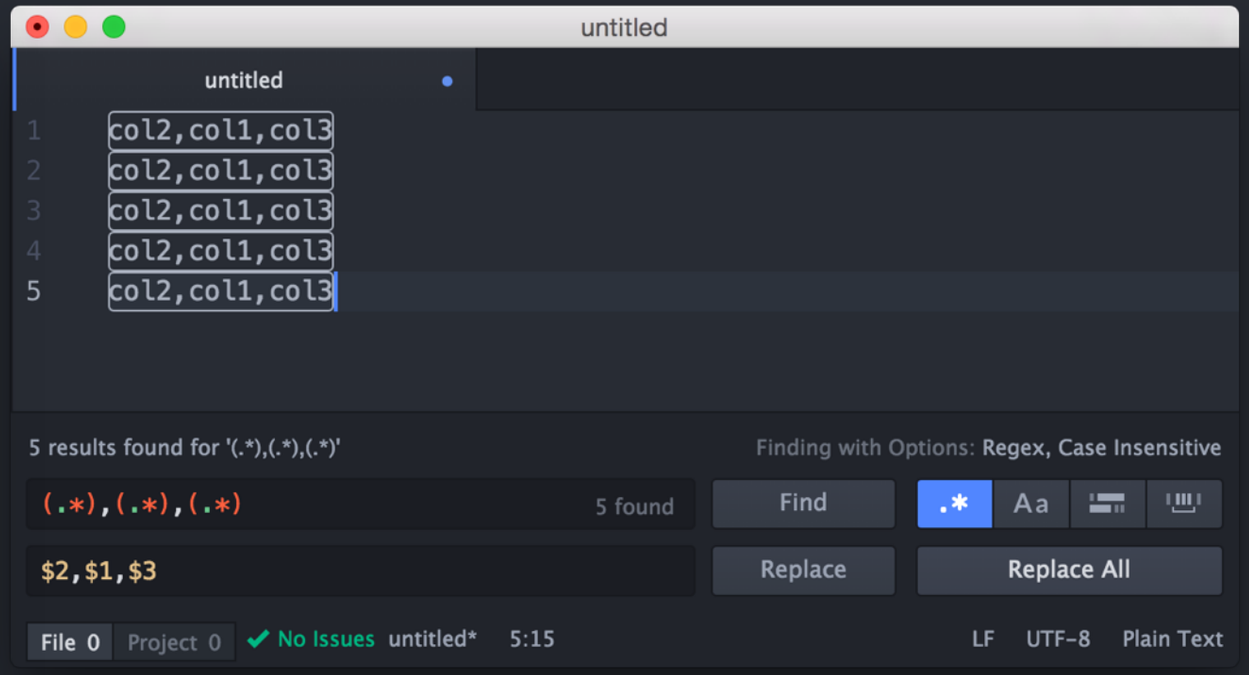
```
1   1. Syntax elements
2
3   \       Escape special char or start a sequence.
4   R|S     Match either regex R or regex S.
5   ()      Create capture group, & indicate precedence
6   []      Enclose a set of matchable chars
7
8   2. Characters
9
10  .       Match any char except newline, see re.DOTALL
11  \w      Alphanumeric: [0-9a-zA-Z_], see LOCALE
12  \W      Non-alphanumeric
13  \s      Whitespace [ \t\n\r\f\v], see LOCALE,UNICODE
14  \S      Non-whitespace
15  \d      Digit
16  \D      Non-digit
17  \xHH    Two digit hexadecimal character goes here
```

Regular Expressions

Why are they **awesome**?

Rearrange columns

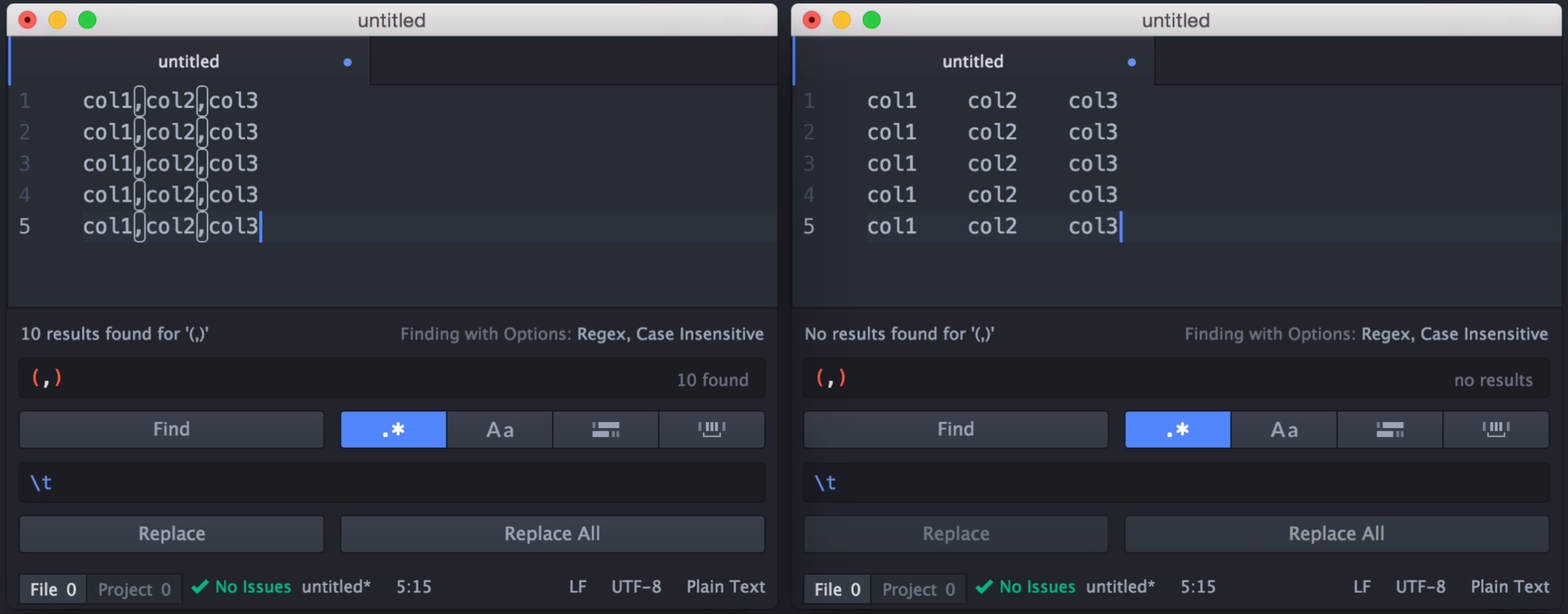




Regular Expressions

Why are they **awesome**?

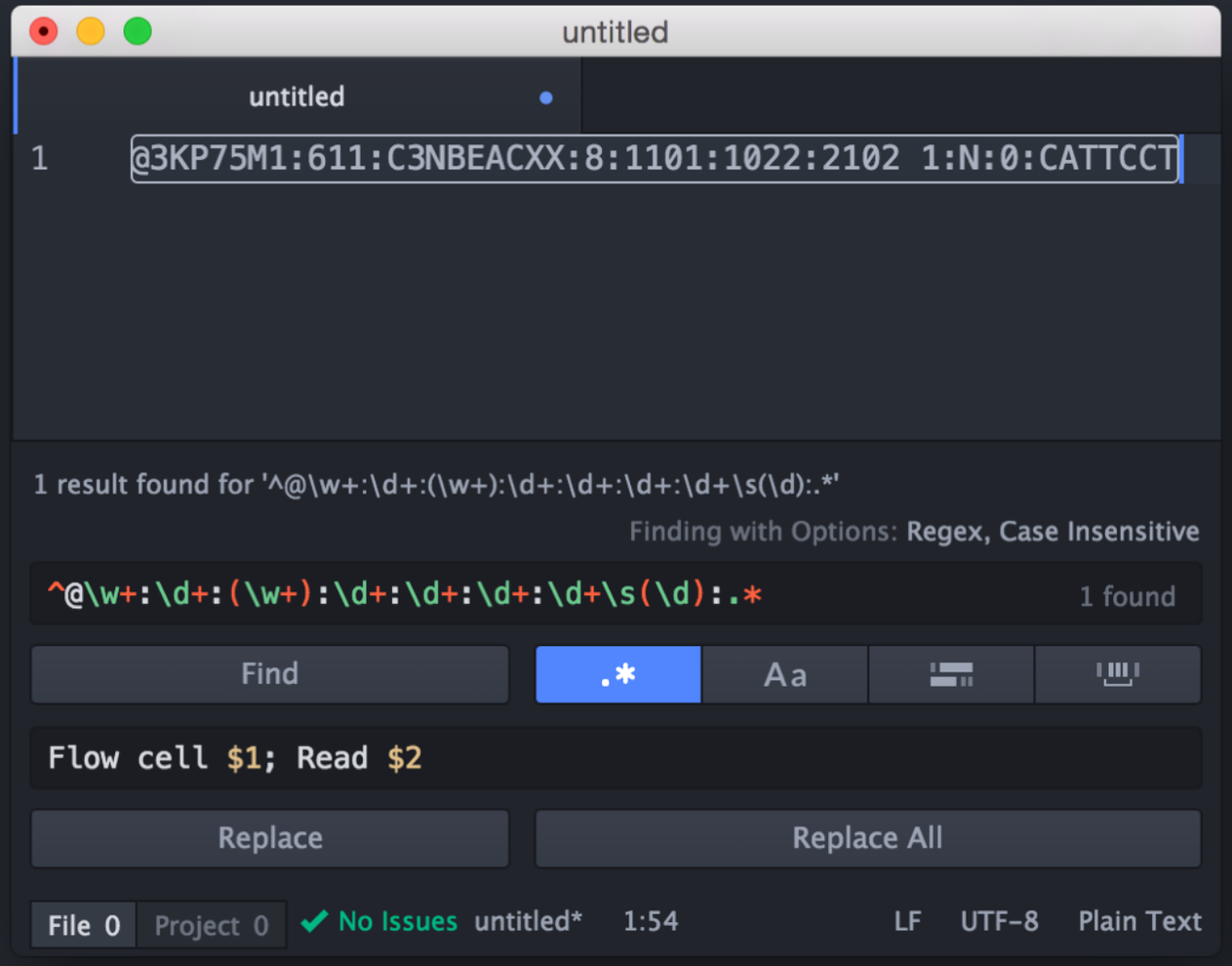
CSV to TDT

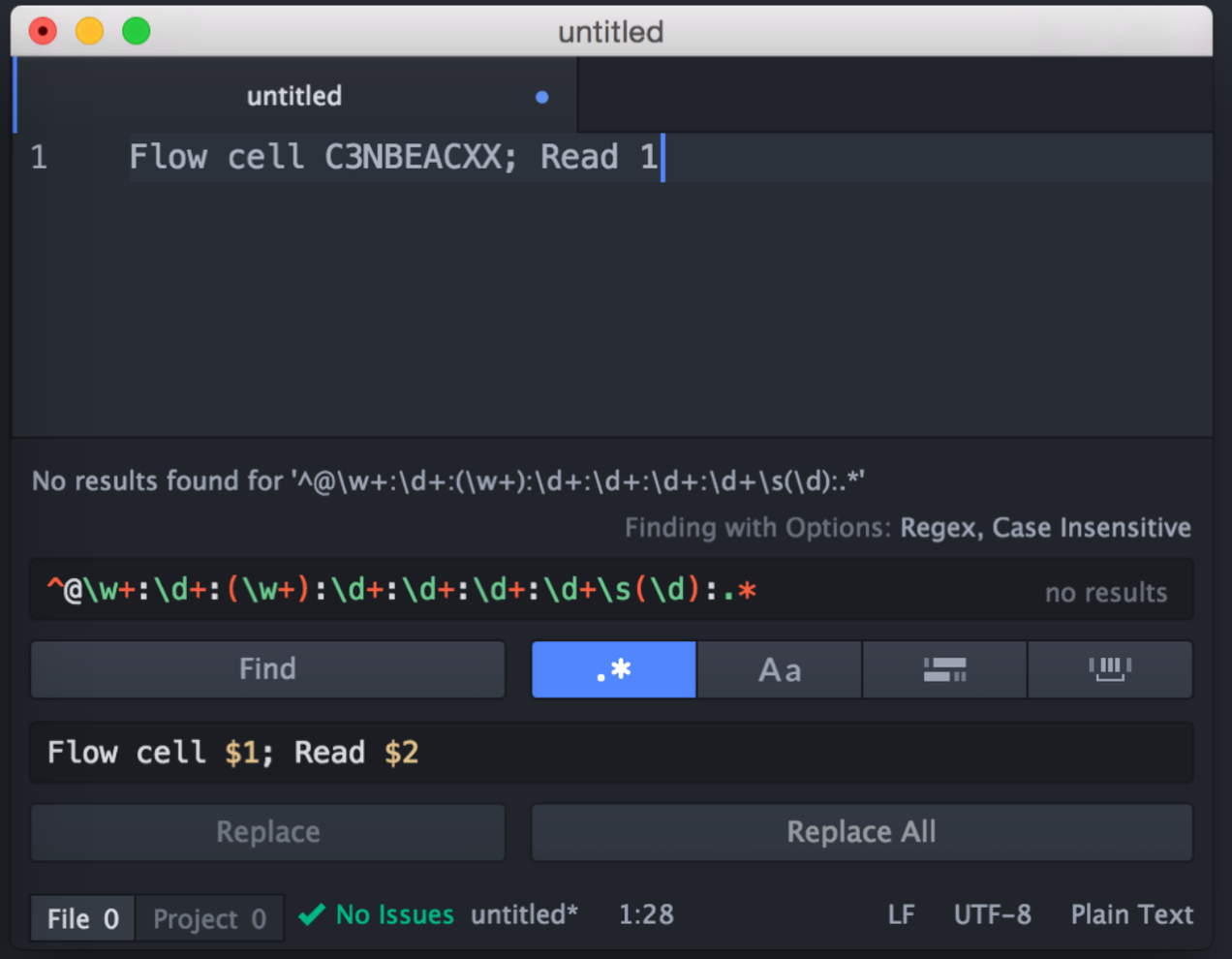


Regular Expressions

Why are they **awesome**?

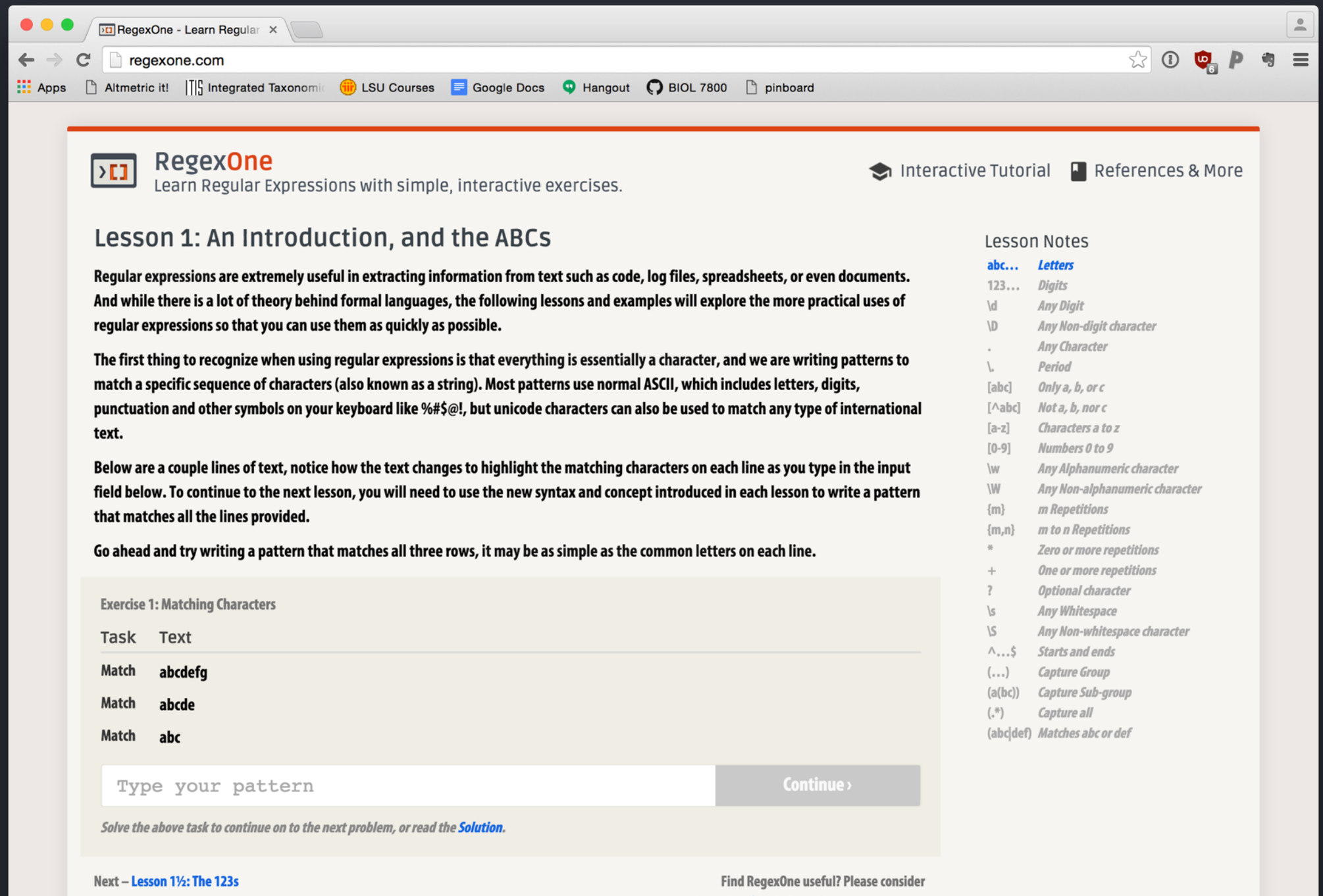
Extracting and Replacing





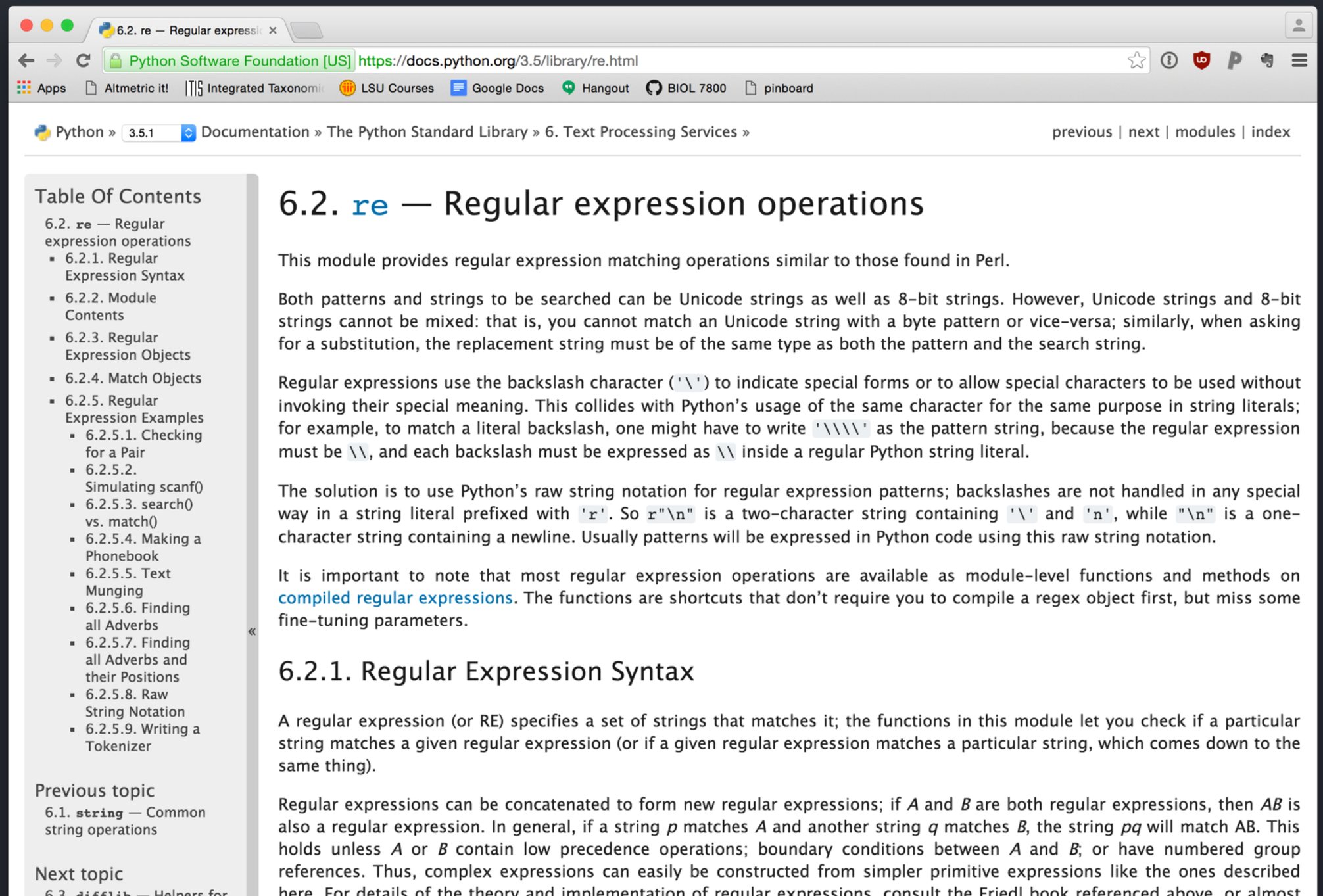
Regular Expressions

How do I learn them?



Regular Expressions

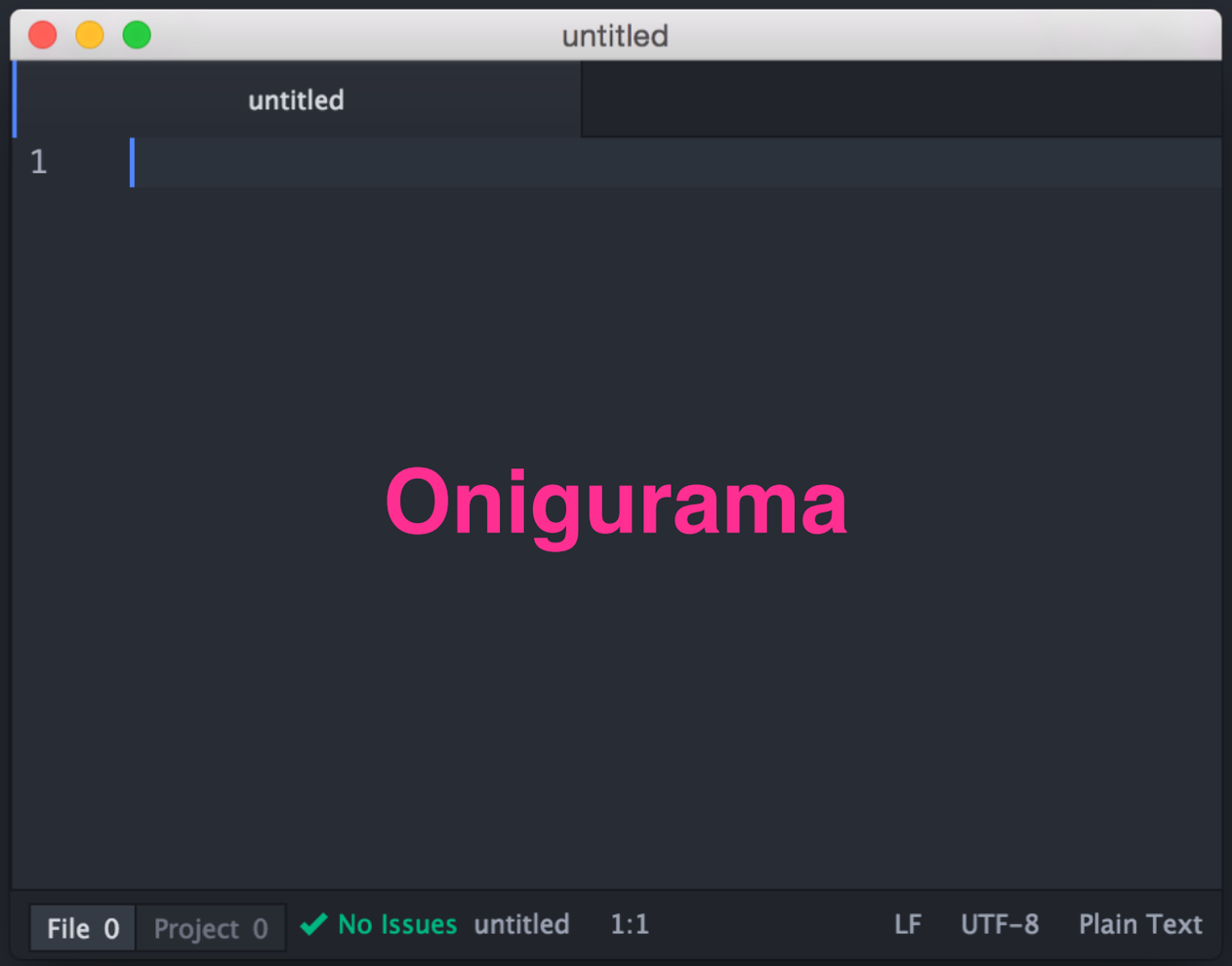
How do I learn them?



Regular Expressions

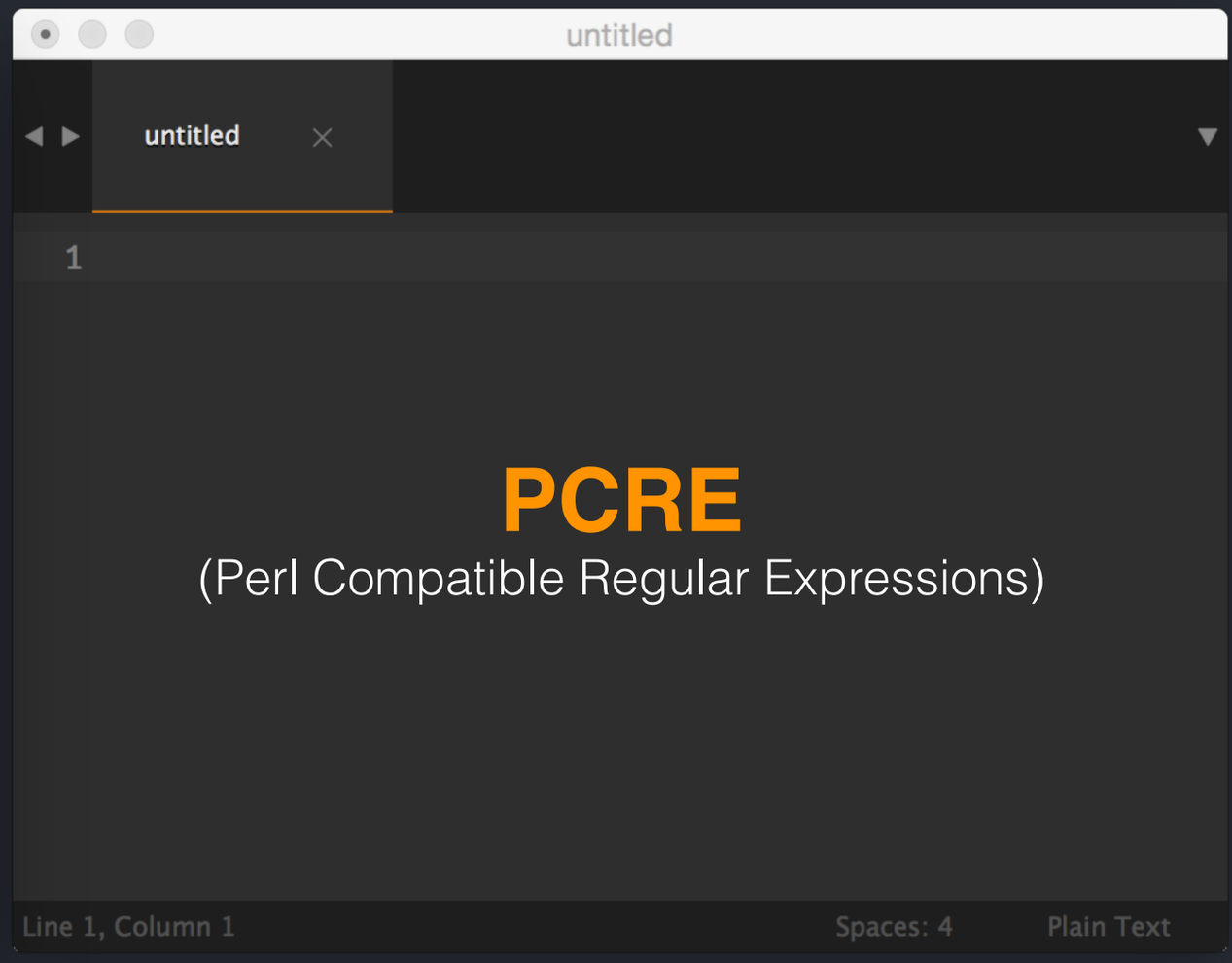
How do I learn them?

Learn *your* regex engine



Oniguruma

oniguruma.rubyforge.org



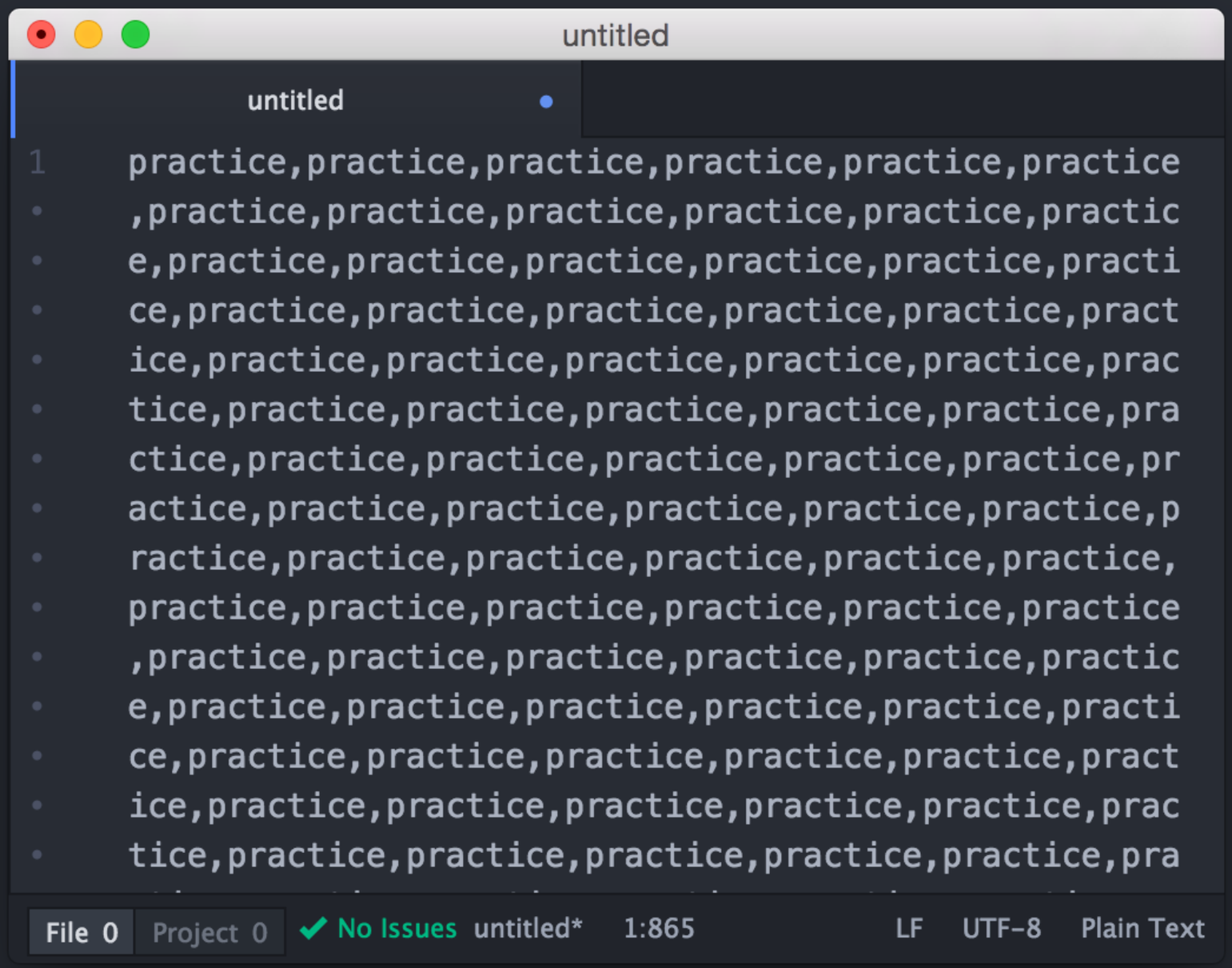
PCRE

(Perl Compatible Regular Expressions)

www.boost.org

Regular Expressions

How do I learn them?



Regular Expressions

…and that's what we're going to do **right now**…