# [320] Regular Expressions

Tyler Caraza-Harter

# Reading

**New text:** *Principles and Techniques of Data Science*
by Sam Lau, Joey Gonzalez, and Deb Nolan

Used for Berkeley's DS100 Course.

**Read Chapter 13**: https://www.textbook.ds100.org/ch/13/text_regex.html

```python
# HIDDEN
def show_regex_match(text, regex):
    """
    Prints the string with the regex match highlighted.
    """
    print(re.sub(f'({regex})', r'\033[1;30;43m\1\033[m', text))
```

```python
# The show_regex_match method highlights all regex matches in the
regex = r"green"
show_regex_match("Say! I like green eggs and ham!", regex)
```
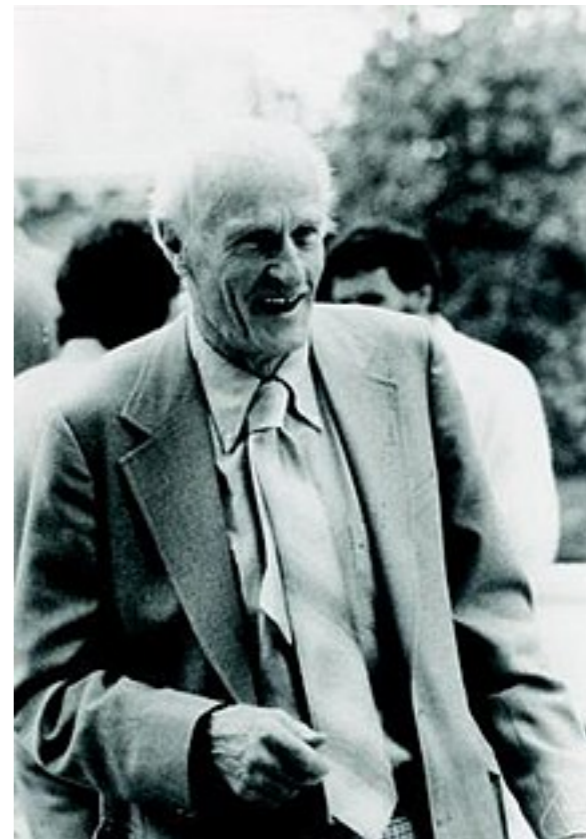
Say! I like green eggs and ham!

be sure to expand
the hidden cells!

# Regular Expressions

Regex:
- a small language for describing patterns to search for
- regex patterns are used in many different programming languages (like how many different languages might use SQL queries)
- https://blog.teamtreehouse.com/regular-expressions-10-languages



Stephen Cole Kleene
(UW-Madison mathematician)

msg = "In CS 320, there are 14 quizzes, 7 projects, 41 lectures, and 1000 things to learn.  CS 320 is awesome!"

```
# does the string contain "320"?
has_320 = msg.find("320") >= 0
```

str.find is VERY limited -- what if we want to:
- find all occurrences of "320"
- find any 3-digit numbers?
- find any numbers at all?
- find a number before the word "projects"?
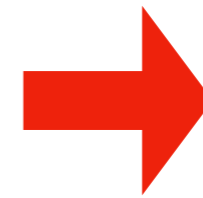- substitute a number for something else?

Regexes can do all these things!

# In Python, regular expressions usually use "raw" strings

what character(s) does `print("A\tB")` print between "A" and "B"?

# In Python, regular expressions usually use "raw" strings

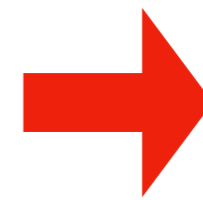what character(s) does `print("A\tB")` print between "A" and "B"?

TAB, because backslash is the escape character

*what if we actually want a backslash and a "t"?*

# In Python, regular expressions usually use "raw" strings

what character(s) does `print("A\tB")` print between "A" and "B"?

*TAB, because backslash is the escape character*

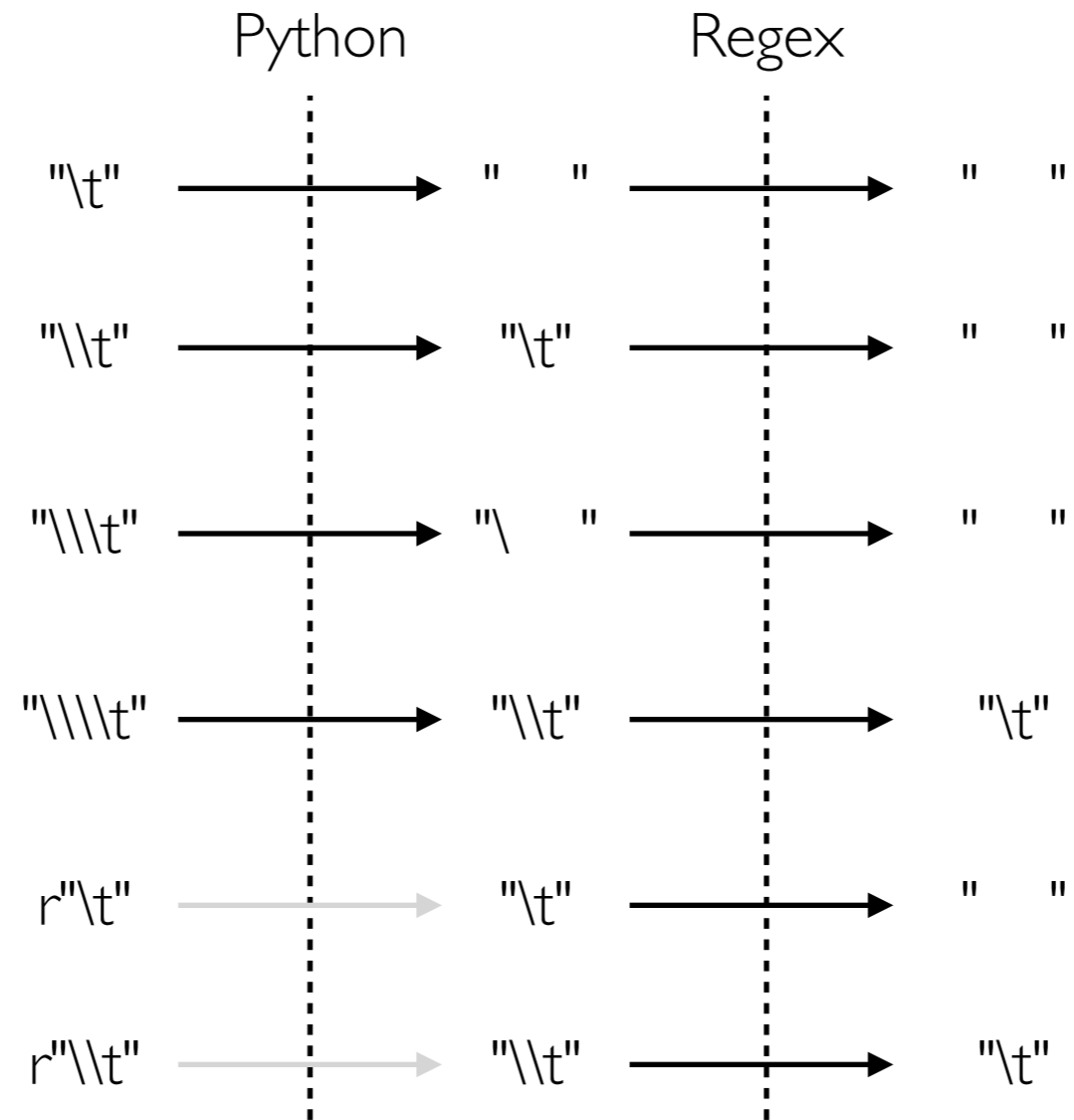*what if we actually want a backslash and a "t"?*

`print("A\\tB")`

`print(r"A\tB")`

*this is a **raw** string, so "\" isn't an escape character*

*Python regex functions do their own escaping, so this is very handy!*

# Double Escaping

# Notebook Demos (copy/paste to start)...

```
import re

# from DS100 book...
def reg(regex, text):
    """
    Prints the string with the regex match highlighted.
    """
    print(re.sub(f'({regex})', r'\033[1;30;43m\1\033[m', text))
s1 = " ".join(["A DAG is a directed graph without cycles.",
            "A tree is a DAG where every node has one parent (except the root, which
has none).",
            "To learn more, visit www.example.com or call 1-608-123-4567. :)
¯\_(ツ)_/¯"])
print(s1)


s2 = """1-608-123-4567
a-bcd-efg-hijg (not a phone number)
1-608-123-456 (not a phone number)
608-123-4567
123-4567
1-123-4567
"""
print(s2)

s3 = "In CS 320, there are 14 quizzes, 7 projects, 41 lectures, and 1000 things to
learn.  CS 320 is awesome!"
print(s3)

s4 = """In CS 320,  there are 14 quizzes,    7 projects,
41 lectures, and 1000 things to learn.  CS 320 is awesome!"""
print(s4)
```
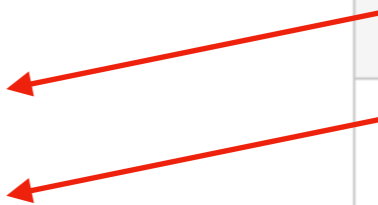
# Learn Regex Features!

Good overview here: https://www.textbook.ds100.org/ch/08/text_regex.html#Reference-Tables

(screenshots here for convenience)

*?

non-greedy equivalents:

+?

| Char | Description | Example | Matches | Doesn't Match |
|---|---|---|---|---|
| . | Any character except \n | . . . | abc | ab abcd |
| [ ] | Any character inside brackets | [cb.]ar | car .ar | jar |
| [^ ] | Any character *not* inside brackets | [^b]ar | car par | bar ar |
| * | ≥ 0 or more of last symbol | [pb]*ark | bbark ark | dark |
| + | ≥ 1 or more of last symbol | [pb]+ark | bbpark bark | dark ark |
| ? | 0 or 1 of last symbol | s?he | she he | the |
| {n} | Exactly *n* of last symbol | hello{3} | hellooo | hello |
| \| | Pattern before or after bar | wel[ui]s | we us is | e s |
| \ | Escapes next character | \[hi\] | [hi] | hi |
| ^ | Beginning of line | ^ark | ark two | dark |
| $ | End of line | ark$ | noahs ark | noahs arks |

| Description | Bracket Form | Shorthand |
|---|---|---|
| Alphanumeric character | [a-zA-Z0-9] | \w |
| Not an alphanumeric character | [^a-zA-Z0-9] | \W |
| Digit | [0-9] | \d |
| Not a digit | [^0-9] | \D |
| Whitespace | [\t\n\f\r\p{Z}] | \s |
| Not whitespace | [^\t\n\f\r\p{z}] | \S |

# Python `re` Module: `findall` and `sub`

```python
import re

s = 'In CS 320, there are 10 quizzes, 7 projects, 39
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```python
re.findall(r"\d+", s)        re.sub(r"\d+", "###", s)
```

pattern   input str       pattern   replacement   input str

# Python `re` Module: `findall` and `sub`

```python
import re

s = 'In CS 320, there are 10 quizzes, 7 projects, 39
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```python
re.findall(r"\d+", s)
```

pattern    input str

```python
re.sub(r"\d+", "###", s)
```

pattern    replacement    input str

```
['320', '10', '7',
'39', '1000', '320']
```

```
'In CS ###, there are ### quizzes, ###
projects, ### lectures, and ### things
to learn.  CS ### is awesome!'
```

# Groups

```
import re

s = 'In CS 320, there are 10 quizzes, 7 projects, 39
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```
re.findall(r"(\d+) (\w+)", s)
```

group 1      group 2

# Groups

```
import re

s = 'In CS 320, there are 10 quizzes, 7 projects, 39
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

```
re.findall(r"(\d+) (\w+)", s)
```

group 1    group 2

```
[('10', 'quizzes'), ('7', 'projects'), ('39', 'lectures'),
 ('1000', 'things'), ('320', 'is')]
```

# Groups

```python
import re

s = 'In CS 320, there are 10 quizzes, 7 projects, 39
lectures, and 1000 things to learn.  CS 320 is
awesome!'
```

group 1

```python
re.findall(r"((\d+) (\w+))", s)
```
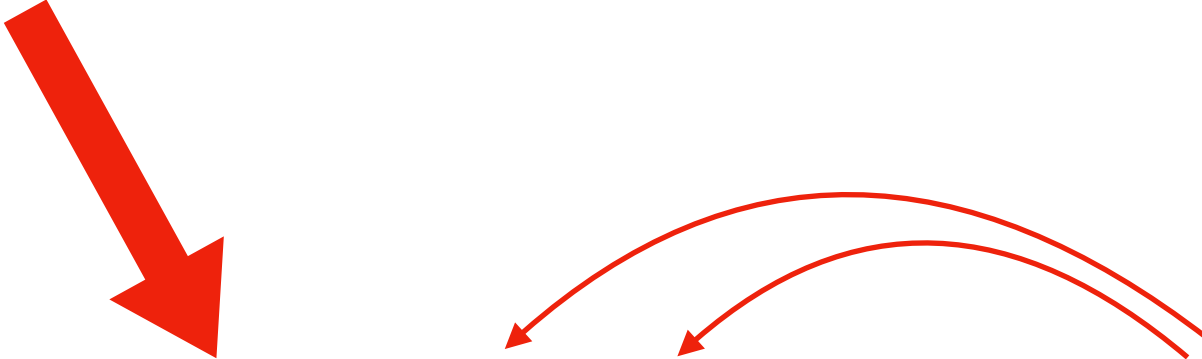
group 2        group 3

```python
[('10 quizzes', '10', 'quizzes'),
 ('7 projects', '7', 'projects'),
 ('39 lectures', '39', 'lectures'),
 ('1000 things', '1000', 'things'),
 ('320 is', '320', 'is')]
```

# Python `re` Module: `findall` and `sub`

```python
import re

s = """In CS 320,  there are 10 quizzes,    7 projects,
41 lectures, and 1000 things to learn.  CS 320 is
awesome!"""
```

2 spaces     tab     newline

```python
re.sub(r"\s+", " ", s)
```

pattern    replacement    input str

single space is
only separator!

```
'In CS 320, there are 10 quizzes, 7 projects,
39 lectures, and 1000 things to learn. CS 320
is awesome!'
```

# Python `re` Module: `findall` and `sub`

```python
import re

s = """In CS 320,  there are 10 quizzes,    7 projects,
41 lectures, and 1000 things to learn.  CS 320 is
awesome!"""
```

```python
re.sub(r"(\d+)", "\<b\>\g<1>\</b\>", s)
```

use \g<N> to refer to group N

# Python `re` Module: `findall` and `sub`

```python
import re

s = """In CS 320,  there are 10 quizzes,    7 projects,
41 lectures, and 1000 things to learn.  CS 320 is
awesome!"""
```

```python
re.sub(r"(\d+)", r"<b>\g<1></b>", s)
```
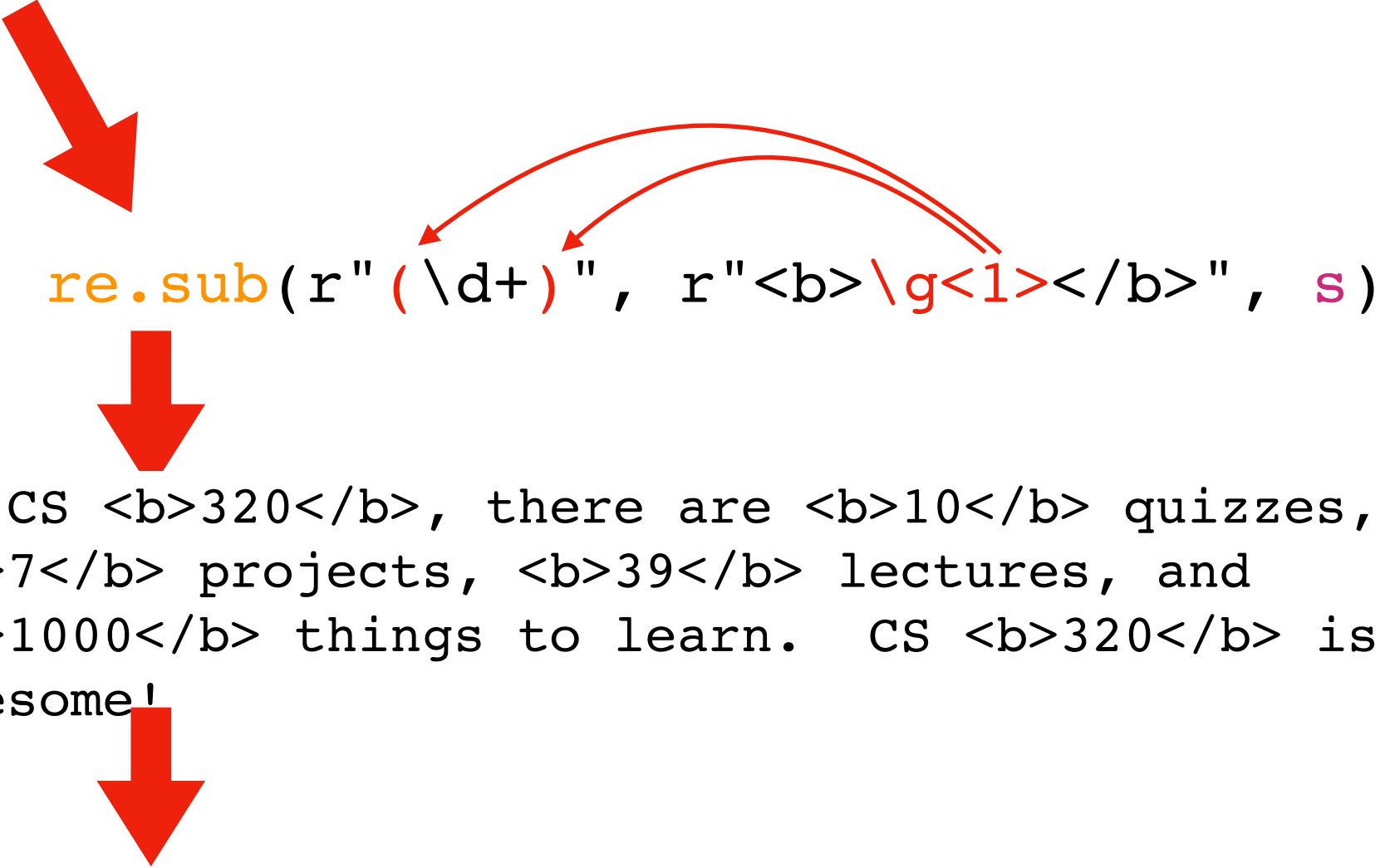
```
In CS <b>320</b>, there are <b>10</b> quizzes,
<b>7</b> projects, <b>39</b> lectures, and
<b>1000</b> things to learn.  CS <b>320</b> is
awesome!
```

In CS **320**, there are **10** quizzes, **7** projects, **39** lectures, and **1000** things to learn. CS **320** is awesome!

# Review Regular Expressions

Which regex will **NOT** match "123"
1. `r"\d\d\d"`
2. `r"\d{3}"`
3. `r"\D\D\D"`
4. `r"..."`

What will `r"^A"` match?
1. `"A"`
2. `"^A"`
3. `"BA"`
4. `"B"`
5. `"BB"`

Which one can match "HH"?
1. `r"HA+H"`
2. `r"HA+?H"`
3. `r"H(A+)?H"`

Which string(s) **will** match `r"^(ha)*$"`
1. `""`
2. `"hahah"`
3. `"that"`
4. `"HAHA"`

What is the type of the following?
`re.findall(r"(\d) (\w+)", some_str)[0]`
1. `list`
2. `tuple`
3. `string`

What will it do?
```
re.sub(r"(\d{3})-(\d{3}-\d{4})",
       r"(\g<1>) \g<2>",
       "608-123-4567")
```

# Practice

finding emails, extracting function names, other examples...