Workshop structure

Intro to Regex

a. What is regex? -> sequence of characters (string) that specifies a match pattern in text
b. Where is regex used? -> pattern matching, data cleaning
c. Introduce regexr.com (use for most of workshop)
d. Basic components (literal char, special char, char classes, anchors, etc.)

Check-in Code: r3g3x

Workshop structure

Practical Applications (Group Activity?)

a. Validate user input: Matching patterns in text (small set of strings–match using regex)
b. Data extraction: Regex puzzle (extract specific data from some structured data)
c. Text processing (python notebook demonstration of regex usage)

Check-in Code: r3g3x

Workshop structure

Quiz/Challenge (kahoot w/ prize?)

a. Questions like:
    i. Which regex matches an email address
    ii. Which regex matches all numbers in a string
    iii. Find the mistake in the following regex

Check-in Code: r3g3x

ACM at UCSD

Workshop structure

Further Learning

a. Introduce advanced regex components (skim over, to show its potential) (lookahead/lookbehind assertions, substitution replacement, flags, etc.)
b. Resources
    i. regexr.com reference page
    ii. regular-expressions.info
    iii. YouTube tutorials
    iv. CSE 105

Check-in Code: r3g3x

Unlocking the Language of Patterns: Intro to Regex

Lloyd Seo

Check-in Code: r3g3x

ACM at UCSD

Check-in Code

for

Unlocking the Language of Patterns: ...

r3g3x

ACM at UCSD

members.acmucsd.com

Icebreaker

Get to know the people sitting around you!

1. name
2. pronouns
3. year
4. college

Check-in Code: r3g3x

ACM at UCSD

Agenda

1. **Introduction**
   What is regex?
2. **Regex Basics**
   Basic components
3. **Regex Quiz**
   Kahoot w/ prizes 👀
4. **Applications**
   Practical uses for regex
5. **Further Learning**
   Advanced components and resources

Check-in Code: r3g3x

ACM at UCSD

Agenda

1. **Introduction**
   What is regex?
2. **Regex Basics**
   Basic components
3. **Regex Quiz**
   Kahoot w/ prizes 👀
4. **Applications**
   Practical uses for regex
5. **Further Learning**
   Advanced components and resources

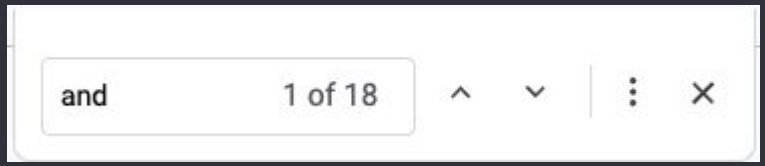
Check-in Code: r3g3x

ACM at UCSD

random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool

Check-in Code: r3g3x

ACM at UCSD

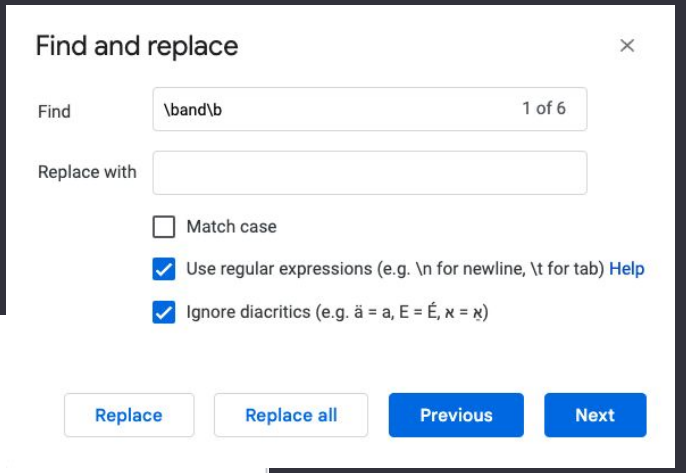


random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool

Check-in Code: r3g3x

ACM at UCSD

“\band\b” -> finds all occurrences of the word “and”



random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool

Check-in Code: r3g3x

ACM at UCSD

Regular Expressions (aka regex)

Sequence of characters that specifies a match pattern in text.

Check-in Code: r3g3x

python

```python
import re

text = "random words and more random words"
pattern = r"\band\b"
matches = re.findall(pattern, text)
print(matches)  # ['and']
```

javascript

```javascript
const text = "random words and more random words";
const pattern = /\band\b/g;
console.log(text.match(pattern));  // ['and']
```

cpp

```cpp
#include <iostream>
#include <regex>
#include <string>

int main() {
    std::string text = "random words and more random words";
    std::regex pattern("\\band\\b");  // Double escaping
    std::smatch matches;

    if (std::regex_search(text, matches, pattern)) {
        std::cout << matches[0] << std::endl;  // "and"
    }
}
```

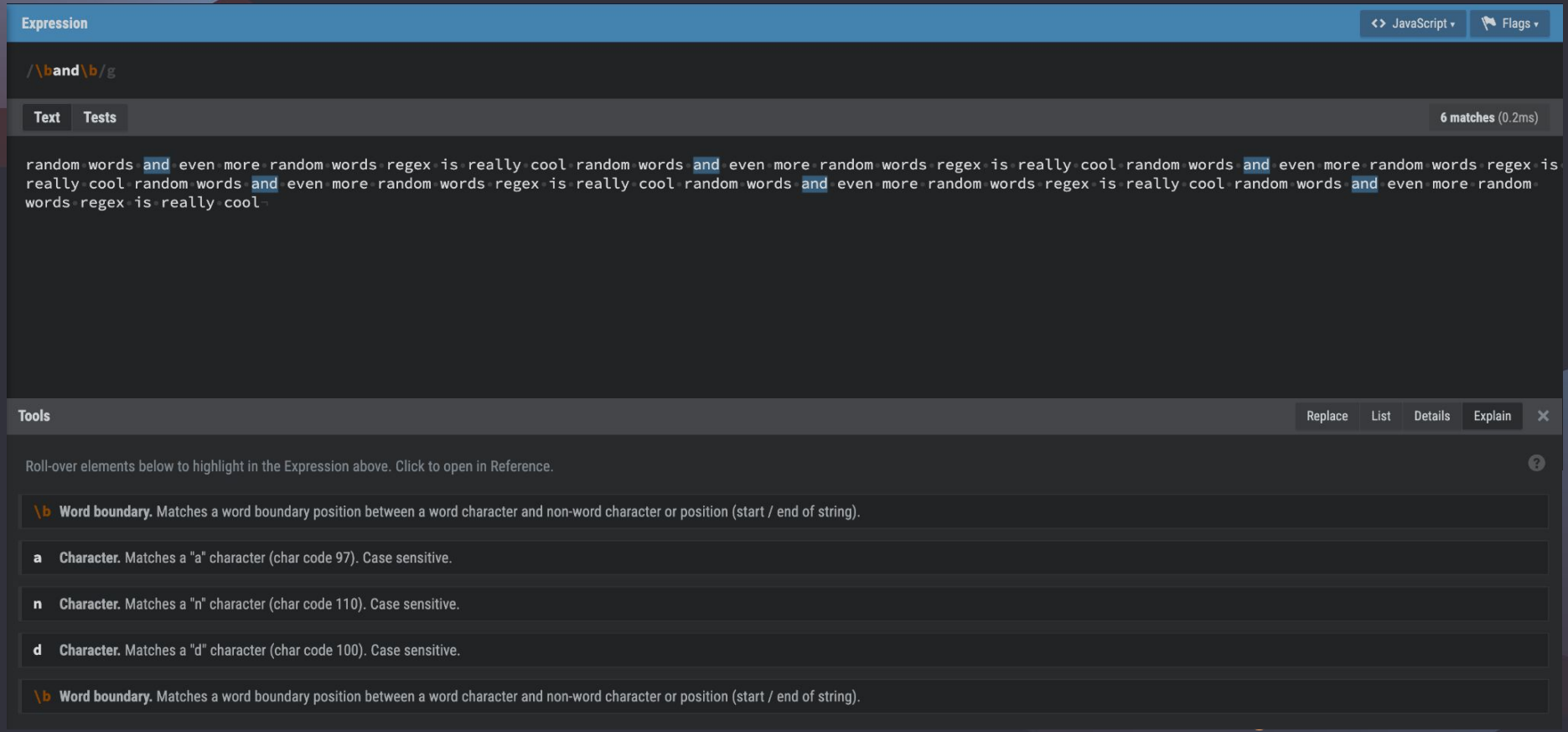
java

```java
import java.util.regex.*;

public class Main {
    public static void main(String[] args) {
        String text = "random words and more random words";
        Pattern pattern = Pattern.compile("\\band\\b");  // Double escaping
        Matcher matcher = pattern.matcher(text);

        while (matcher.find()) {
            System.out.println(matcher.group());  // "and"
        }
    }
}
```

regexr.com


Expression
JavaScript
Flags
/\band\b/g
Text
Tests
6 matches (0.2ms)
random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool random words and even more random words regex is really cool
Tools
Replace
List
Details
Explain
Roll-over elements below to highlight in the Expression above. Click to open in Reference.
\b Word boundary. Matches a word boundary position between a word character and non-word character or position (start / end of string).
a Character. Matches a "a" character (char code 97). Case sensitive.
n Character. Matches a "n" character (char code 110). Case sensitive.
d Character. Matches a "d" character (char code 100). Case sensitive.
\b Word boundary. Matches a word boundary position between a word character and non-word character or position (start / end of string).

Agenda

1. **Introduction**
   What is regex?
2. **Regex Basics**
   Basic components
3. **Regex Quiz**
   Kahoot w/ prizes 👀
4. **Applications**
   Practical uses for regex
5. **Further Learning**
   Advanced components and resources

Check-in Code: r3g3x

ACM at UCSD

Basic Metacharacter Types

Classes: [ABC], [A-Z], ., \w, \d, \s

Anchors: ^, $, \b

Quantifiers: +, *, {*i*, *j*}, ?, ?

Groups: (ABC), \1, |

Escaped Characters: \+, \t, \n

Check-in Code: r3g3x

ACM at UCSD

Characters

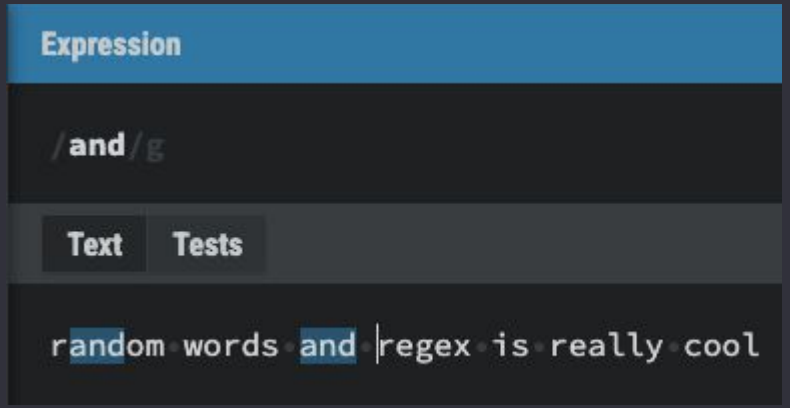
Literal characters that become part of the match pattern:

- Letters
- Numbers
- Certain punctuation marks ($, !, etc.)

Check-in Code: r3g3x

ACM at UCSD

Characters



Without additional metacharacters, these function the same as Ctrl + F/Cmd + F

Check-in Code: r3g3x

ACM at UCSD

Characters

Literal characters that become part of the match pattern:

- Letters
- Numbers
- Certain punctuation marks (!, @, #, etc.)

Check-in Code: r3g3x

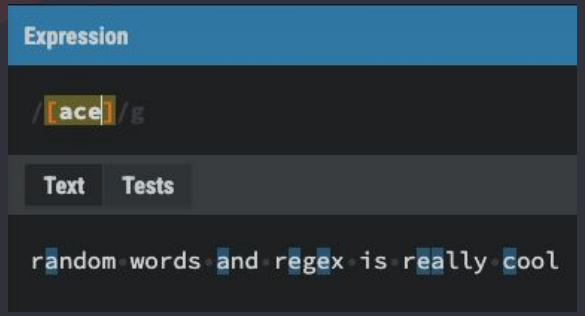
Classes

Matches a character from a set:

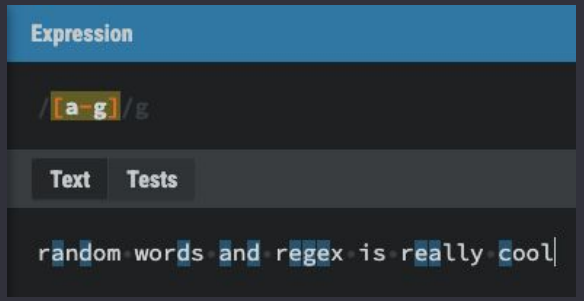
- `[ABC]` (char set): Any character in set
- `[A-Z]` (range): Any character within range of set
- `.` (wildcard): Any character except newline

Check-in Code: r3g3x

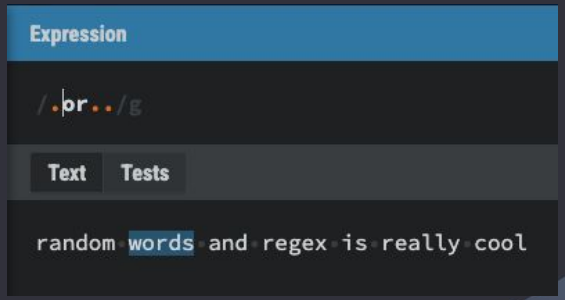
Classes



All characters in set



All characters in range specified by set



All characters except newline

Check-in Code: r3g3x

ACM at UCSD

Classes

Q. What would [abc0-9] match?

Check-in Code: r3g3x

Classes

Q. What would [abc0-9] match?

A. Matches a, b, c, and all numbers

Check-in Code: r3g3x

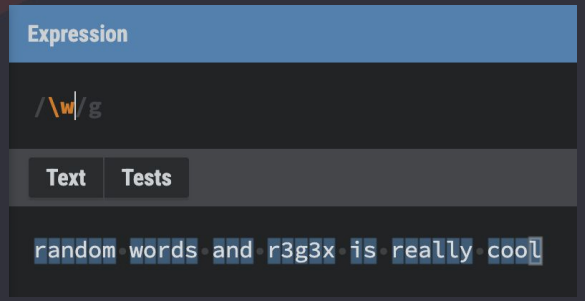
Classes

Matches a character from a set:

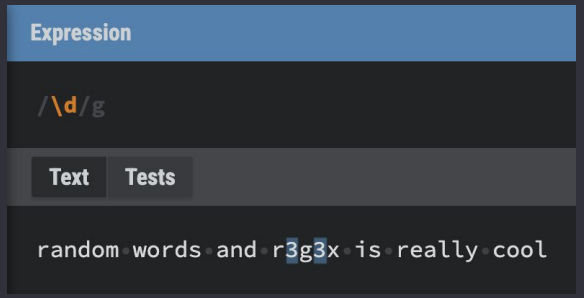
- `\w` (word): Any word character
- `\d` (digit): Any digit character
- `\s` (whitespace): Any whitespace

Check-in Code: r3g3x

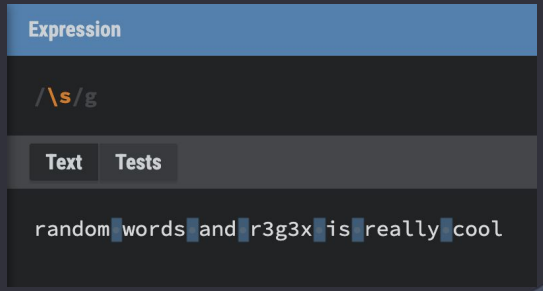
Classes



All word characters



All characters in range specified by set



All characters except newline

Check-in Code: r3g3x

ACM at UCSD

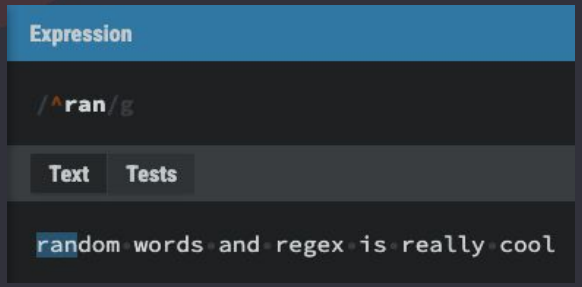
Anchors

Matches a position (instead of a character):

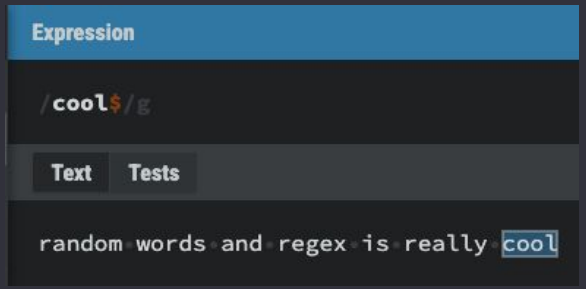
- `^` (beginning): Beginning of string
- `$` (end): End of string
- `\b` (word boundary): Position between word and non-word characters

Check-in Code: r3g3x

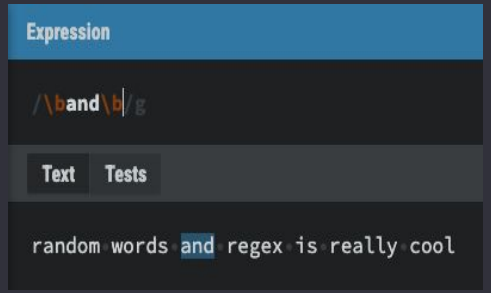
Anchors



Beginning of string



End of string



Position between word and non-word characters

Check-in Code: r3g3x

ACM at UCSD

Anchors

Q: What would ^hi\b match?

Check-in Code: r3g3x

Anchors

Q: What would ^hi\b match?

A: Matches beginning of string if it is exactly “hi”

(ex. Can’t be hint, hit, etc.)

Check-in Code: r3g3x

ACM at UCSD

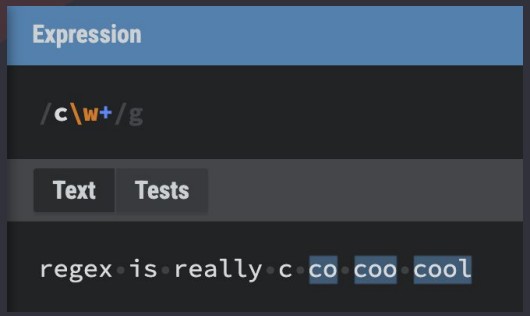
Quantifiers

Matches preceding character repeatedly:

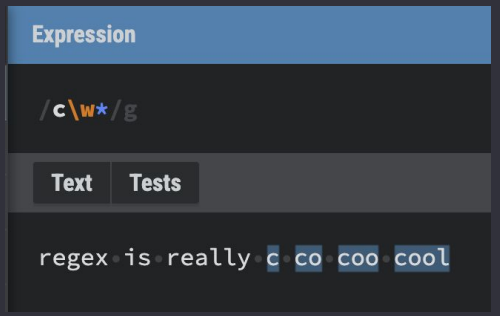
- `+` (plus): Matches at least 1 or more of preceding character
- `*` (star): Matches 0 or more of preceding character
- `{i, j}` (quantifier): Matches *i* to *j* of preceding character

Check-in Code: r3g3x

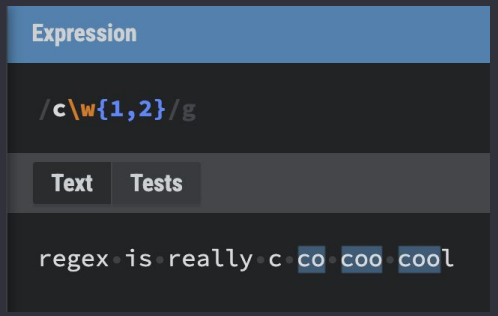
Quantifiers



All word characters



All characters in range specified by set



All characters except newline

Check-in Code: r3g3x

ACM at UCSD

Quantifiers

Q: How many a's in a row would a{1,2} match?

Check-in Code: r3g3x

Quantifiers

Q: How many a's in a row would a{1,2} match?

A: 1 or 2 a's

Check-in Code: r3g3x

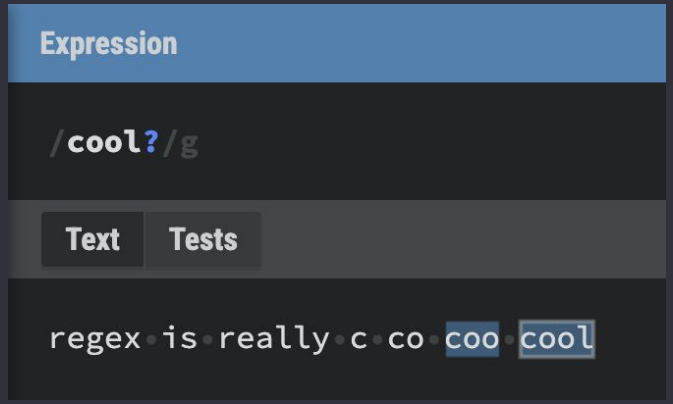
Quantifiers

Matches preceding character repeatedly:

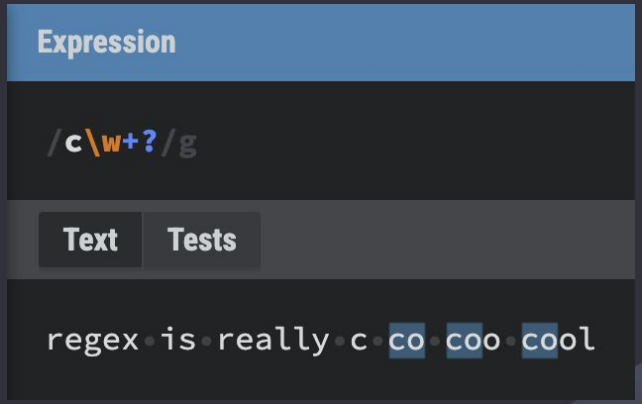
- `?` (optional): Makes preceding character optional
- `?` (lazy): Matches preceding character as few times as possible

Check-in Code: r3g3x

Quantifiers



Makes preceding character optional



Matches preceding character as few times as possible

Check-in Code: r3g3x

ACM at UCSD

Quantifiers

Q: Which of the following would ap?ple match?

- apple
- aple
- ale

Check-in Code: r3g3x

Quantifiers

Q: Which of the following would ap?ple match?

- apple
- aple
- ale

A: 2, would match apple and aple

Check-in Code: r3g3x

ACM at UCSD

Groups

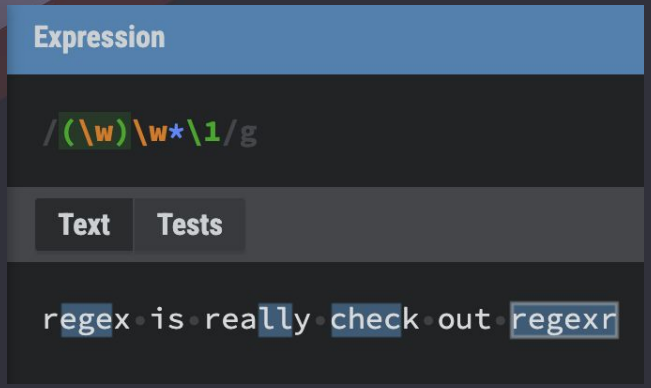
Allows grouping characters to be processed together:

- (ABC) (capture group): Groups characters together and creates a capture group
    - Capture groups are numbered groups you can access
- \1 (backreference): Matches results of corresponding capture group
- | (alternation): Acts like a boolean OR

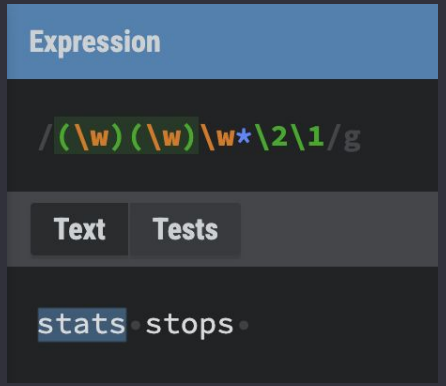
Check-in Code: r3g3x

ACM at UCSD

Groups



Groups and creates a capture group



Refer to multiple capture groups

Acts like boolean OR

Check-in Code: r3g3x

ACM at UCSD

Groups

Q: Which of the following would (a)(b)\2\1 match?

- **baba**
- **abba**
- **abab**
- **ab**
- **ba**

Check-in Code: r3g3x

Groups

Q: Which of the following would (a)(b)\2\1 match?

- **baba**
- **abba**
- **abab**
- **ab**
- **ba**

A: abba

Check-in Code: r3g3x

ACM at UCSD

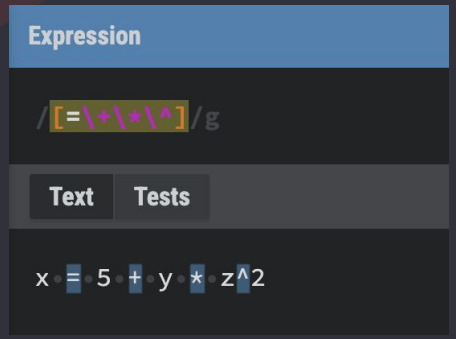
Escaped Characters

Match reserved/special characters/metacharacters:

- `\+` (reserved characters): Matches +*?^$\.[]{}()|/
- `\t` (tab): Matches tab characters
- `\n` (newline): Matches newline characters

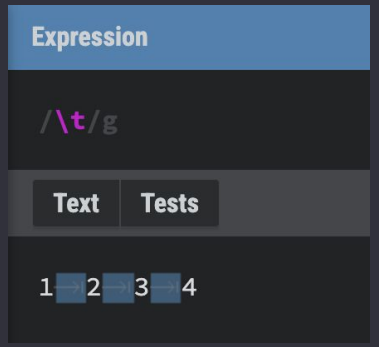
Check-in Code: r3g3x

Escaped Characters



Matches

+*?^$\.[]{}()|/



Matches tab character

All characters except newline

Check-in Code: r3g3x

ACM at UCSD

Escaped Characters

Q: Which of the following would .\.. Match?

- a.b
- ...
- .a.

Check-in Code: r3g3x

Escaped Characters

Q: Which of the following would .\.. Match?

- a.b
- ...
- .a.

A: a.b and ...

Check-in Code: r3g3x

ACM at UCSD

Basic Metacharacter Types

Classes: [ABC], [A-Z], ., \w, \d, \s

Anchors: ^, $, \b

Quantifiers: +, *, {*i*, *j*}, ?, ?

Groups: (ABC), \1, |

Escaped Characters: \+, \t, \n

Check-in Code: r3g3x

ACM at UCSD

Agenda

1. **Introduction**
   What is regex?
2. **Regex Basics**
   Basic components
3. **Regex Quiz**
   Kahoot w/ prizes 👀
4. **Applications**
   Practical uses for regex
5. **Further Learning**
   Advanced components and resources

Check-in Code: r3g3x

ACM at UCSD

Kahoot!

Top three will get ACM raccoon tote bags!

Check-in Code: r3g3x

ACM at UCSD

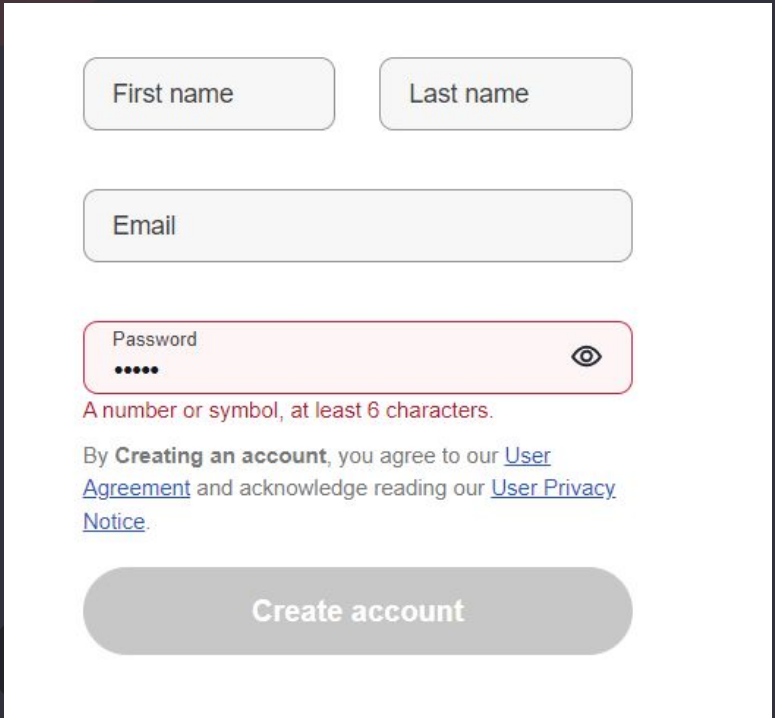
Agenda

1. **Introduction**
   What is regex?
2. **Regex Basics**
   Basic components
3. **Regex Quiz**
   Kahoot w/ prizes 👀
4. **Applications**
   Practical uses for regex
5. **Further Learning**
   Advanced components and resources

Check-in Code: r3g3x

ACM at UCSD

Applications of Regex



Input validation:

- Name input (match all alphabets)
- Email (validate format)
- Password (check requirements)

Applications of Regex

A	B
Name	Legal status
Bob	citizen
Kevin	resident
Emily	non-immigrant
Ryan	non-immigrant
Tiffany	citizen

Data extraction:

- Only match applications where the legal status is “citizen” or “resident”

Check-in Code: r3g3x

ACM at UCSD

Applications of Regex

```
1. <div class="product"><h2>Apple iPhone 13</h2><p>Price: <span class="pr
2. <div class="product"><h2>Samsung Galaxy S21</h2><p>Price: <span class=
3. <div class="product"><h2>Google Pixel 6</h2><p>Price: <span class="pri
```

Data cleaning:

- Remove HTML tags from dataset
- Remove extra whitespaces

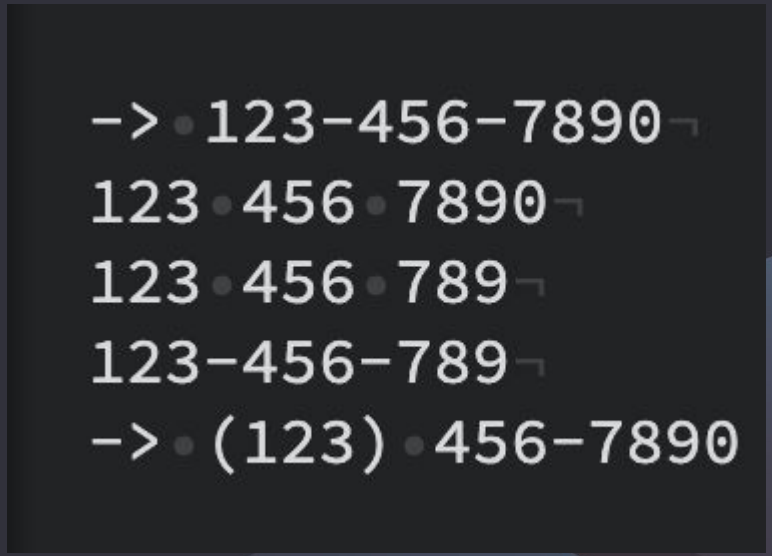
Check-in Code: r3g3x

ACM at UCSD

Activity 1: Phone Number

We can identify valid phone number formats. How can we match the following formats?:

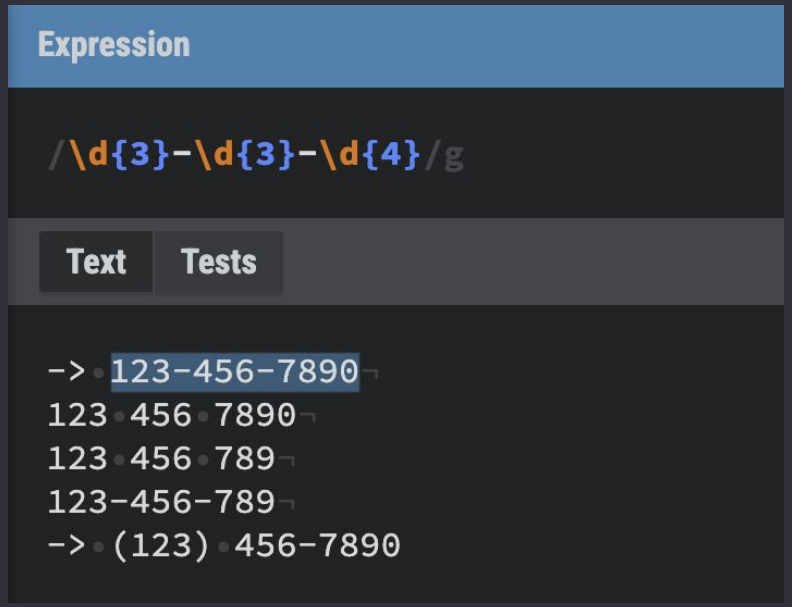
- 123-456-7890 (hint: classes and quantifiers)
- (123) 456-7890



Check-in Code: r3g3x

ACM at UCSD

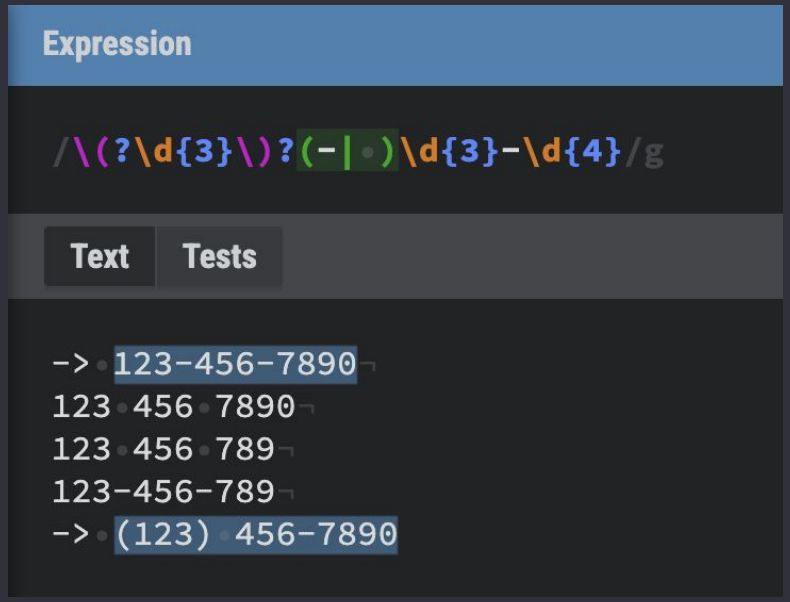
Match First Format



Check-in Code: r3g3x

ACM at UCSD

Match Both Formats



Check-in Code: r3g3x

ACM at UCSD

Activity 2: Email Address

How can we extract email addresses from a text?

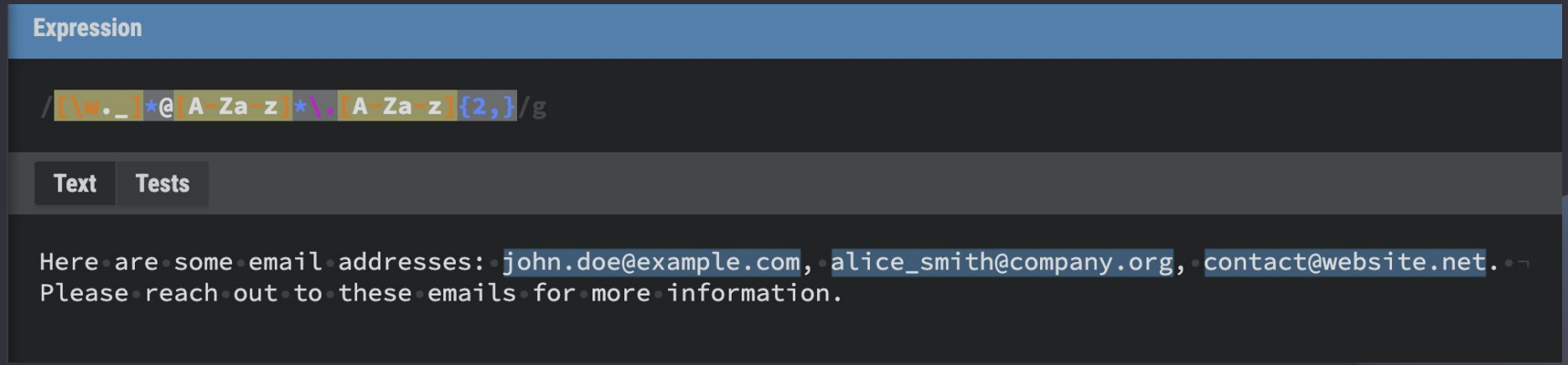
- *(alphanumeric, period, underscore)@(alphabets).(alphabets >= 2)*

Hint: classes, quantifiers, escaped character

Check-in Code: r3g3x

ACM at UCSD

Match Email Address Format



(alphanumeric, period, underscore)@(alphabets).(alphabets >= 2)

Check-in Code: r3g3x

ACM at UCSD

Agenda

1. **Introduction**
   What is regex?
2. **Regex Basics**
   Basic components
3. **Regex Quiz**
   Kahoot w/ prizes 👀
4. **Applications**
   Practical uses for regex
5. **Further Learning**
   Advanced components and resources

Check-in Code: r3g3x

ACM at UCSD

Advanced Metacharacters

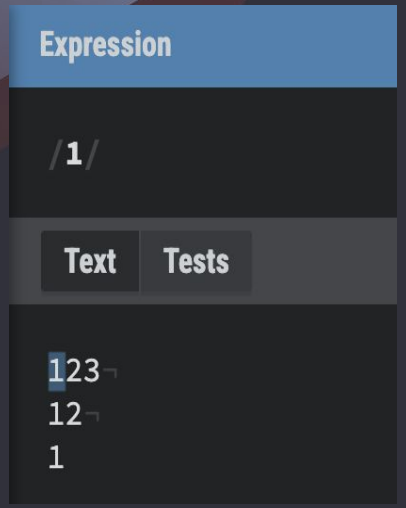
Flags: Changes how the expression is interpreted

- **/g (global search):** Expression match all instances
- **/i (ignore case):** Expression becomes case-insensitive
- **/m (multiline):** Each line becomes its own string

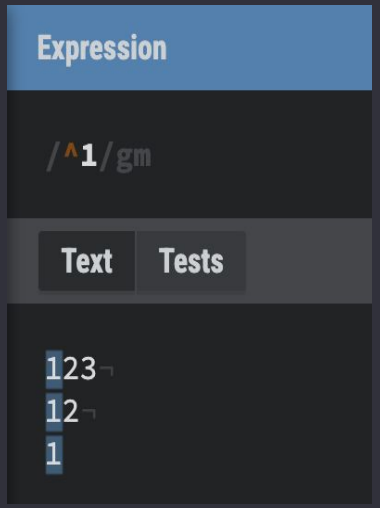
`/regex/g`

Check-in Code: r3g3x

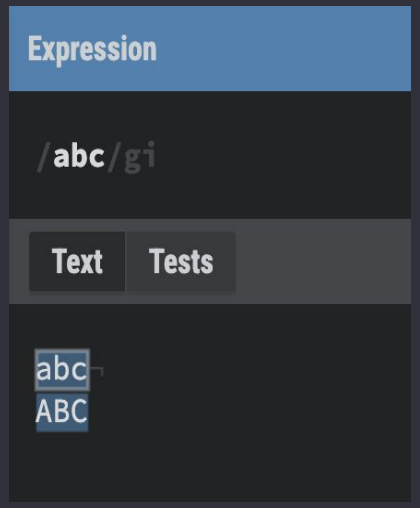
Flags



Without \g, regex only matches the first instance



With \m, each line becomes its own string



With \i, regex becomes case-insensitive

Check-in Code: r3g3x

ACM at UCSD

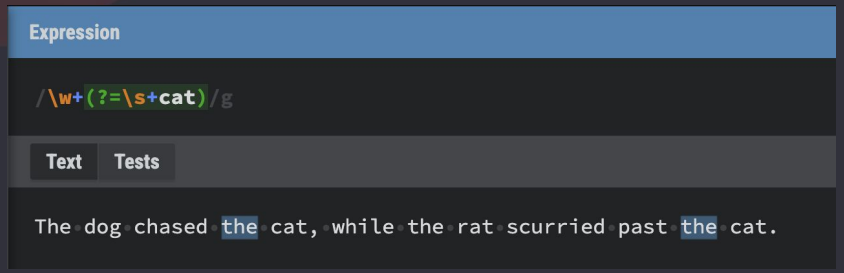
Advanced Metacharacters

Lookahead: Matches a group before the match pattern

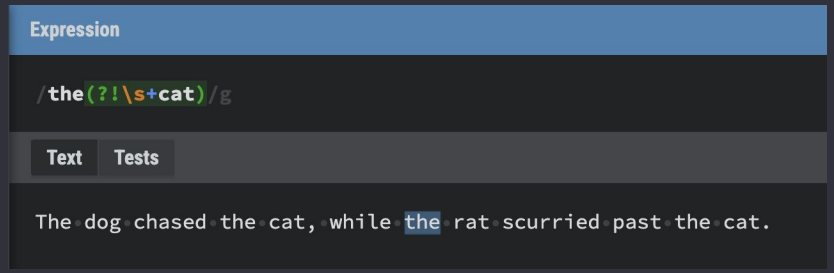
- (?=ABC) (positive lookahead):
  Asserts certain pattern must follow current position
- (?!ABC) (negative lookahead): Asserts certain pattern must not follow current position

Check-in Code: r3g3x

Lookahead



Positive lookahead: matches all word characters that are followed by \s+cat



Negative lookahead: matches all instances of “the” that are NOT followed by \s+cat

Check-in Code: r3g3x

ACM at UCSD

Advanced Metacharacters

Other metacharacters/metacharacter types include:

- Lookbehind
- Substitutions
- Custom classes

Check-in Code: r3g3x

Resources

- regexr.com menu sidebar
- acmurl.com/regex-info
- acmurl.com/regex-python
- Leetcode – Regex problems

Check-in Code: r3g3x

Thank You

Do you have any questions?

contact@acmucsd.org

acmurl.com/**discord**

acmurl.com/**instagram**

acmurl.com/**youtube**

Check-in Code: r3g3x

ACM at UCSD