

# Discussion 11

Regular Expressions 🎉 and SQL 🦴

Antonio Kam

`anto [at] berkeley [dot] edu`

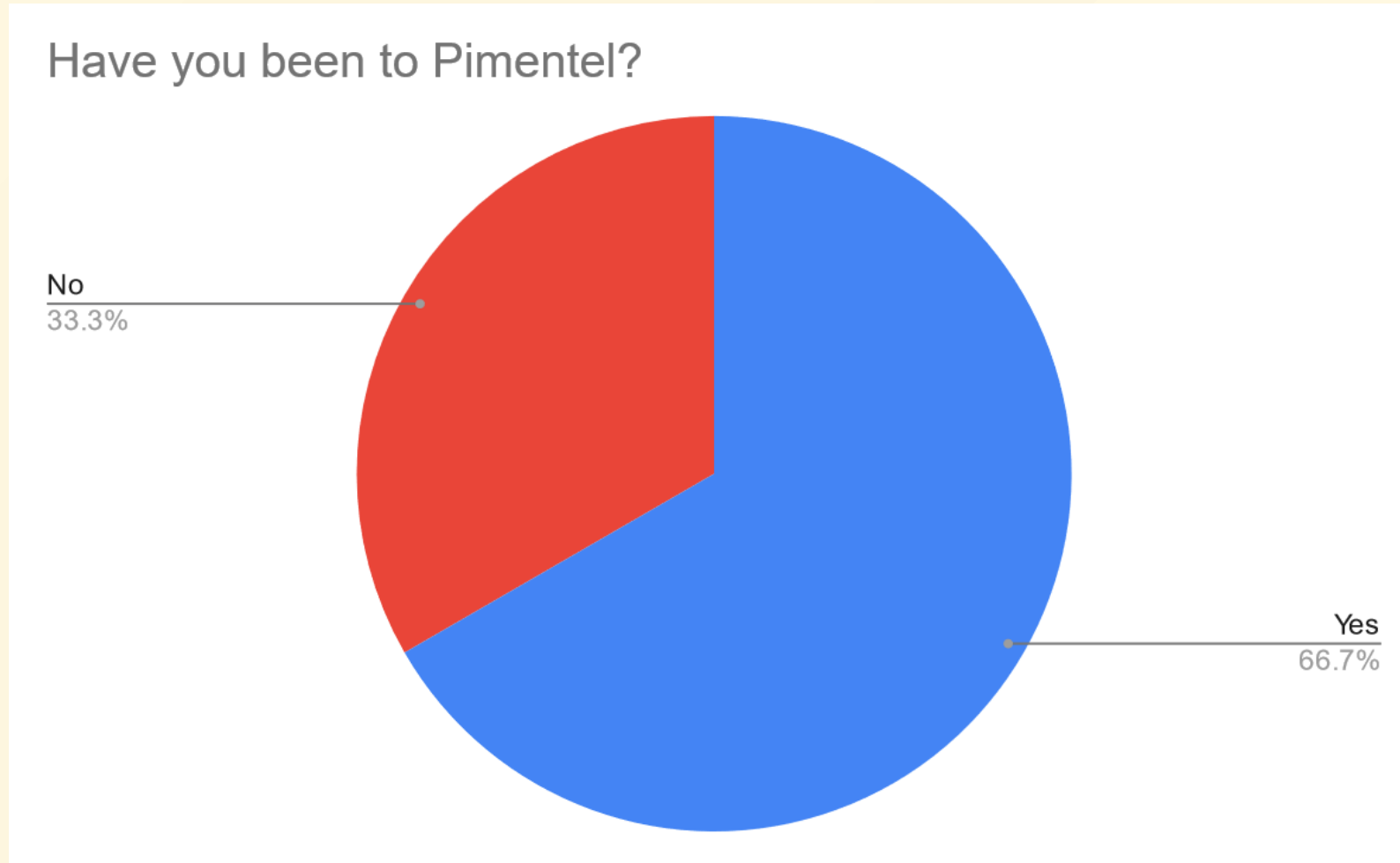
**All slides can be found on**

**[teaching.roux1.es](http://teaching.roux1.es)**

# Announcements

- Scheme Project Tail Recursion (check spec/discussion)
- Teaching opportunities!
  - CSM
  - AI
- Regular Schedule of HW being due on Thursday
- Scheme
  - Checkpoint 1 due today!
  - Checkpoint 2 due Friday 🙄

# Results from last section



# Notes from last section

- What is my morning routine?
- What is my night routine?
- Good humor/energy
  - 🤡

# Temperature Check

- RegEx
- SQL

# RegEx

**Poll: How do we all pronounce it?**

# RegEx

- **Regular Expressions** are used to describe sets of strings that meet certain criteria
- Very useful for pattern matching (will see examples later)
  - Used in real life all the time!
  - Can be used for super simple patterns ( `egg` in text)
  - Can also be very complicated (validating passwords)
- [regex101.com](https://regex101.com)
  - Super useful for learning/debugging your regular expressions, but don't rely on this!
  - (You won't have this resource for the exam! Don't use it as a GPS)



# Character Classes

- If you want to search for a certain character in a set of characters, use character classes
- Can use pre-defined classes, or specify your own

Specified:

Pattern	Description
<code>[abc]</code>	Matches either a, b, or c
<code>[a-z]</code>	Matches anything from lowercase a to lowercase z
<code>[^a-z]</code>	Matches anything that isn't from lowercase a to lowercase z
<code>[0-9]</code>	Matches any (single) digit
<code>[a-zA-Z123]</code>	Matches any lowercase or uppercase character, or the digits 1, 2, or 3

# Character Classes

Predefined character classes

Pattern	Equivalent
<code>\w</code>	<code>[A-Za-z0-9_]</code>
<code>\d</code>	<code>[0-9]</code>
<code>\s</code>	(Matches any 'whitespace' character)
<code>.</code>	Matches everything except for newlines.

# Combining Multiple Patterns

- If you put a pattern after another pattern, you can have a regular expression that matches both of those characters in succession
  - `AB` - Matches A, then B (both are required)
  - `A[xt]e` Matches A, then either x or t, then E
    - `Axe` or `Ate` are the things that get matched
- `|` is the separator used to match 'or'
  - `A|B` - Matches either A or B

# Quantifiers

Quantifier	Description
<code>?</code>	0 or 1 (kind of like a question - is it there or not?)
<code>*</code>	0 or more
<code>+</code>	1 or more
<code>{a}</code>	Match exactly <code>a</code> times (number)
<code>{a, }</code>	Match from <code>a</code> to infinity number of times
<code>{0, b}</code>	Match from <code>0</code> to <code>b</code> number of times
<code>{a, b}</code>	Match between <code>a</code> and <code>b</code> number of times

# Groups

- Parentheses are used to group certain patterns together
  - Useful with quantifiers
  - `(Ha)+` and `Ha+` do different things!

# Anchors

Anchor	Description
<code>^</code>	Matches the <i>beginning</i> of the string
<code>\$</code>	Matches the <i>end</i> of a string
<code>\b</code>	Matches a <i>word boundary</i>

# Special Characters

Some characters cannot be written directly because they have special roles in RegEx

- `\[ ] ( ) { } + * ? | $ ^ .`
- Inside a character class, you need to escape `-`
- To escape these characters, you need to put a backslash before them
  - `The quick brown fox jumped over the lazy dog\.`

# RegEx in Python

- For Python, we use *r-strings* to write regex patterns
- With *r-strings* in Python, backslashes are treated specially in some cases (for example with `\n`, or `\t`). What *r-strings* do in this case is that they automatically escape the backslashes for these special characters, so they look like real backslashes rather than what Python designates for those characters.
- `r"<insert string here>"` - the `r` at the front makes a string an *r-string*
  - Python also has *f-strings* which are pretty useful/interesting



# Worksheet!

# Attendance

[links.roux1.es/disc](https://links.roux1.es/disc)

# SQL



# Select Statements and Queries

# Select Statements

- You can experiment with all of this on [sql.cs61a.org](http://sql.cs61a.org)
- If we have a pre-existing table (for example, the `records` table), we can grab values from that table using a `FROM` clause
- Using `*` will select all columns from a table

```
SELECT [columns] FROM [tables] WHERE [condition] ORDER BY [criteria]
```

Demo:

```
SELECT * FROM records WHERE title = "Programmer";  
SELECT name, salary FROM records WHERE division = "Accounting"  
ORDER BY salary DESC;
```

# Worksheet!

# Joins



# Joins

- Sometimes, people might store data in multiple tables
- It's hard to access data from both these tables
- That's where *joins* come in!

```
SELECT * FROM records, meetings; -- can select multiple tables
```



# Ambiguous Joins

- Tables might have overlapping column names
- We need a way to distinguish between these columns
  - Especially if you need to join a table with itself (useful if you want to compare 2 people with each other)
- Use the `as` keyword

# Ambiguous Joins

```
SELECT a.name, a.title FROM records AS a, records AS b
WHERE a.name = "Louis Reasoner" AND a.supervisor = b.name;
```

# Worksheet!

# Aggregation



# Aggregation

- Aggregation tends to be useful when you have multiple groups, and you want to group by certain pieces of data.
- You can also combine multiple rows into 1 with aggregation
  - `SELECT COUNT(*) FROM RECORDS;`
  - `SELECT name, MAX(salary) FROM RECORDS;`
- `GROUP BY` will allow you to perform these aggregation functions on specific groups
  - `SELECT division, MIN(salary) FROM records GROUP BY division;`
- `WHERE` statements for `GROUP BY`s uses the `HAVING` clause
  - `HAVING` filters out entire groups
  - You can have both `WHERE` and `HAVING` in the same statement

# Worksheet!

# Mental Health Resources

- CAPS:
  - If you need to talk to a professional, please call CAPS at 510-642-9494.
- After Hours Assistance
  - For any assistance after hours, details on what to do can be found at [this link](#)

# Anonymous Feedback Form

[links.roux1.es/feedback](https://links.roux1.es/feedback)

Thanks for coming! 🎉

*Please give me feedback on what to improve!*