List Comprehensions and Data Privacy

- Topics covered in this lecture are:
  - Loop - continue, break, and else
  - List Comprehensions
  - Data Privacy and security concerns

Your future in CS

I used to include this on my slides, but since these slides have changed - going to just leave it up here for every notebook. I get a lot of questions about more programming courses, the concentrations, and minors in computer science. Here is a brief reminder.

CS 164 – Next Course In Sequence, also consider CS 220 (math and stats especially)

- CO Jobs Report 2021 – 77% of *all* new jobs in Colorado require programming
- 60% of all STEM jobs requires *advanced* (200-300 level)
- 31% of all Bachelor of Arts degree titled jobs also required coding skills
- 2016 Report found on average jobs that require coding skills paid $22,000 more

- Concentrations in CS:

  - Computer science has a number of concentrations.
    - General concentration is the most flexible, and even allows students to double major or minor pretty easily.
    - Software Engineering
    - Computing Systems
    - Human Centered Computing
    - Networks and Security
    - Artificial Intelligence
    - Computer Science Education.
  - Minors:
    - Minor in Computer Science - choose your own adventure minor
    - Minor in Machine Learning - popular with stats/math, and engineering
    - Minor in Bioinformatics - Biology + Computer Science

Loops and more Lists

Let's practice more with loops and lists.

Warmup Activity

- Go to the interactive slides

- Write a method that takes in a list of numbers
  - it returns a new list of only the *even* numbers in the list

In [ ]:
```
def evens_only(lst):
    new_lst = []
    for item in lst:
        if item % 2 == 0: new_lst.append(item)
    return new_lst


evans = evens_only([12, 22, 13, 12, 15, 42, 21, 19])
print(evans)
```

```
[12, 22, 12, 42]
```

List Comprehensions

- A very "python" way or programming
- Builds a list of items based on a condition
- Let's try to build a list of random values

In [ ]:
```
import random

old_way = []
for i in range(0, 10):
    old_way.append(random.randint(1,6))
print(old_way)

## in one line
new_way = [random.randint(1,6)  for i in range(0, 10)]
print(new_way)
```

```
[4, 2, 6, 4, 2, 2, 3, 6, 1, 2]
[5, 1, 3, 1, 4, 2, 3, 3, 3, 1]
```

- **[** value to add **for** loop condition **]**
- You can also include an condition to 'add' to the list you are building

In [ ]:
```
odds = [i for i in new_way if i % 2 == 1]
evans = [i for i in new_way if i % 2 == 0]

print(odds)
print(evans)
```

```
[5, 1, 3, 1, 3, 3, 3, 1]
[4, 2]
```

Let's return to converting a String to an int

In [ ]:
```
values = [['rain', '50', 'april'],
          ['snow', '12', 'dec']]

temperatures = [int(i[1]) for i in values]  ## so a line to pull out the temps, buildi
```

```
print(temperatures)
print(sum(temperatures)/len(temperatures))
```

```
[50, 12]
31.0
```

In Class Practice Two

- Take a list of items,
  - Return only the items that have “a” in the word
    - or better yet, use the variable "letter"
  - Use a list comprehension!
  - The *in* keyword will help with the if statement
    - `if "a" in val:`

```
["hello", "alice", "jade", "zypher", "ariel"]
#The new list would be
["alice", "jade", "ariel"]
```

In [ ]:
```
def contains_letter(lst, letter='a'):
    return [val for val in lst if letter in val]

print(contains_letter(["hello", "alice", "jade", "zypher", "ariel"]))
print(contains_letter(["hello", "alice", "jade", "zypher", "ariel"], 'i'))
print(contains_letter(["hello", "alice", "jade", "zypher", "ariel"], 'e'))
```

```
['alice', 'jade', 'ariel']
['alice', 'ariel']
['hello', 'alice', 'jade', 'zypher', 'ariel']
```

Other Useful Commands for Loops

- break
  - ends the loop early
- continue
  - return to the top of the loop, incrementing it
- else
  - code to run if the loop is never ran

In [ ]:
```
# [23, 13, 17, 191]
odds = []
for val in [23, 12, 13, 16, 18, 2, 23, 0, 17, 20, 191]:
    if val in odds: continue
    if val % 2 == 1:
        odds.append(val)
print(odds) #prints odds that only appear once
```

```
[23, 13, 17, 191]
```

In [ ]:
```
# [23, 13, 17, 191]
odds = []
for val in [23, 12, 13, 16, 18, 2, 23, 0, 17, 20, 191]:
    if val in odds:
        break  ## stops running after the first dupplicate
```

```
    if val % 2 == 1:
        odds.append(val)
print(odds)  # prints odds until the first duplicate is found
```

```
[23, 13]
```

In [ ]:
```
val = 10

while val < 5:
    print("This won't execute")
else:
    print("This executes because the loop never executed")
```

```
This executes because the loop never executed
```

Overall

- More useful tools, but technically not required.
- The more you program, the more 'elegant' your code can become
  - these tools also help reduce the amount of code you need

Data Privacy

Let's talk about data. We have a lot of it!

Quick Poll

Which social services do you have an account on?

- A. Facebook
- B. Twitter
- C. TikTok
- D. Linkedin
- E. Other

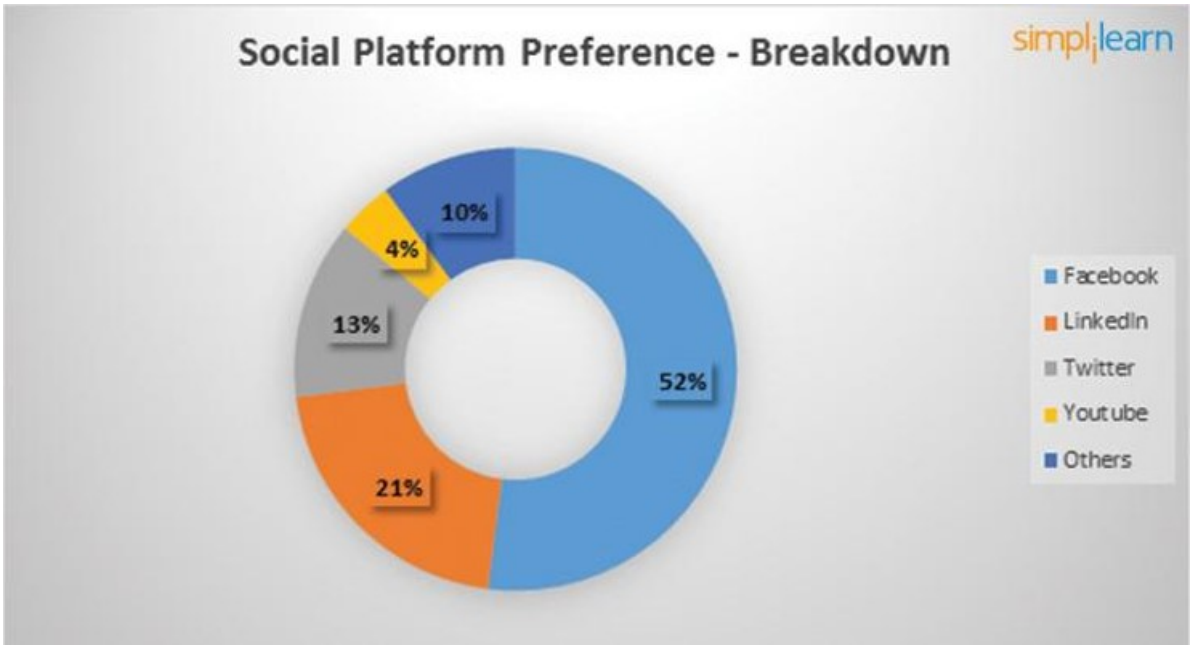
> Discussion
> How much do you use it daily? (some phones you can look this up in settings). Most people know their information is out there, how much do you think about it?

Next Poll

- A. Its out there, and I don't care
- B. Its out there, and little we can do
- C. I try very hard to 'hide' what is out there
- D. Nothing is out there, I don't use anything

> Discussion 2 Thinking about how the internet works, do you feel this information is secure between your computer and your social service? Ok, you may all think

> the correct answer is NO (actually, it is more secure than it may seem), but now ask yourself WHY based on what you understand on networking.



- Facebook
  - 2.45 billion monthly active users
  - 1.62 billion people on average log onto Facebook daily
  - Every 60 seconds on Facebook: 510,000 comments are posted, 293,000 statuses are updated, and 136,000 photos are uploaded.
  - 300 petabyte data analysis warehouse (to start)
- What do they track! (everything)
  - Likes
  - Faces / Images
  - Cookies, tagging, etc
- "Your" Data Worth $240 a year, per person...

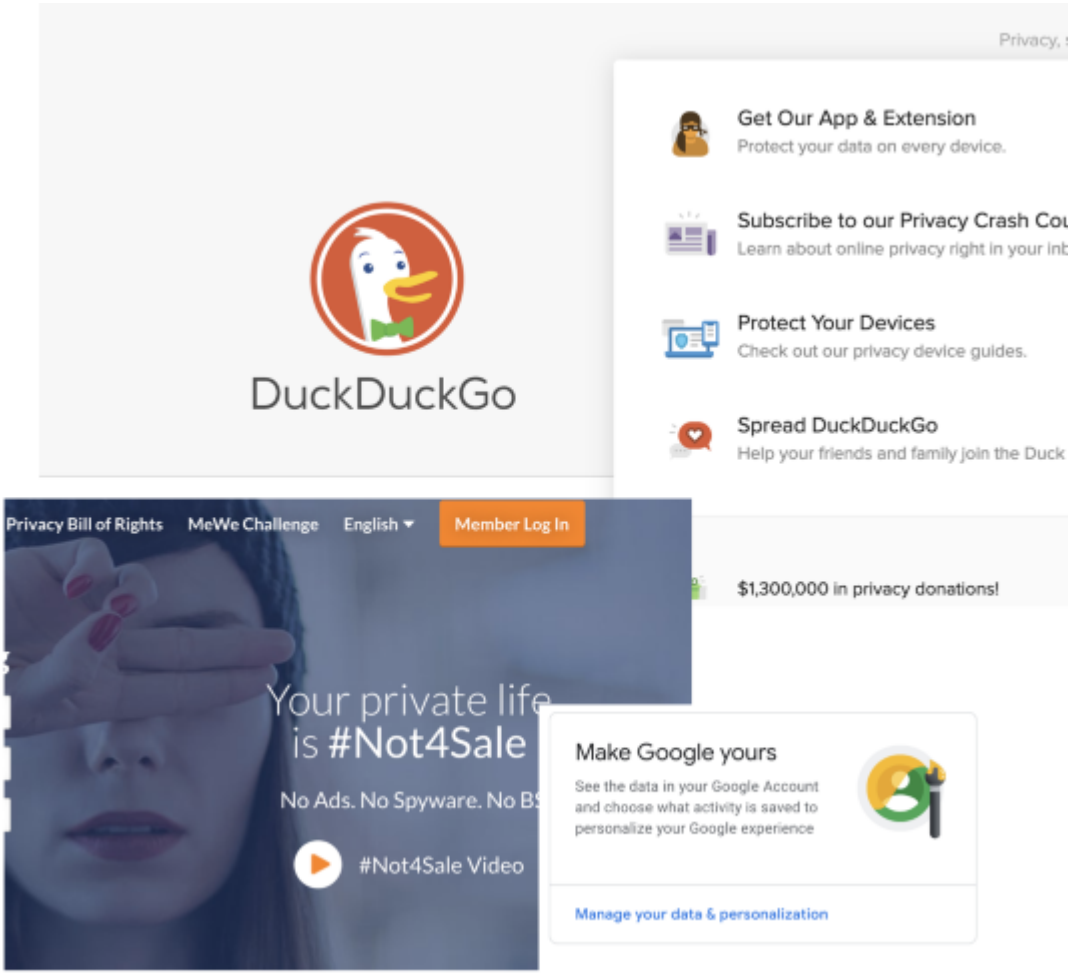
Cookies

- The internet is considered a **stateless** protocol!
- So how does a website know you have visited that website?
  - Cookies
  - Small files stored on your computer
  - These files have information about you
    - Designed to help websites retain information between visits
    - Also used in advertising

```
{"name":"apond", "login-approved":true, "favorite-color":"Red"}
```

- Above is an example cookie (note, most are encrypted now)
- Does the format seem familiar?

- It can have other formats, but these dictionary like strings are common



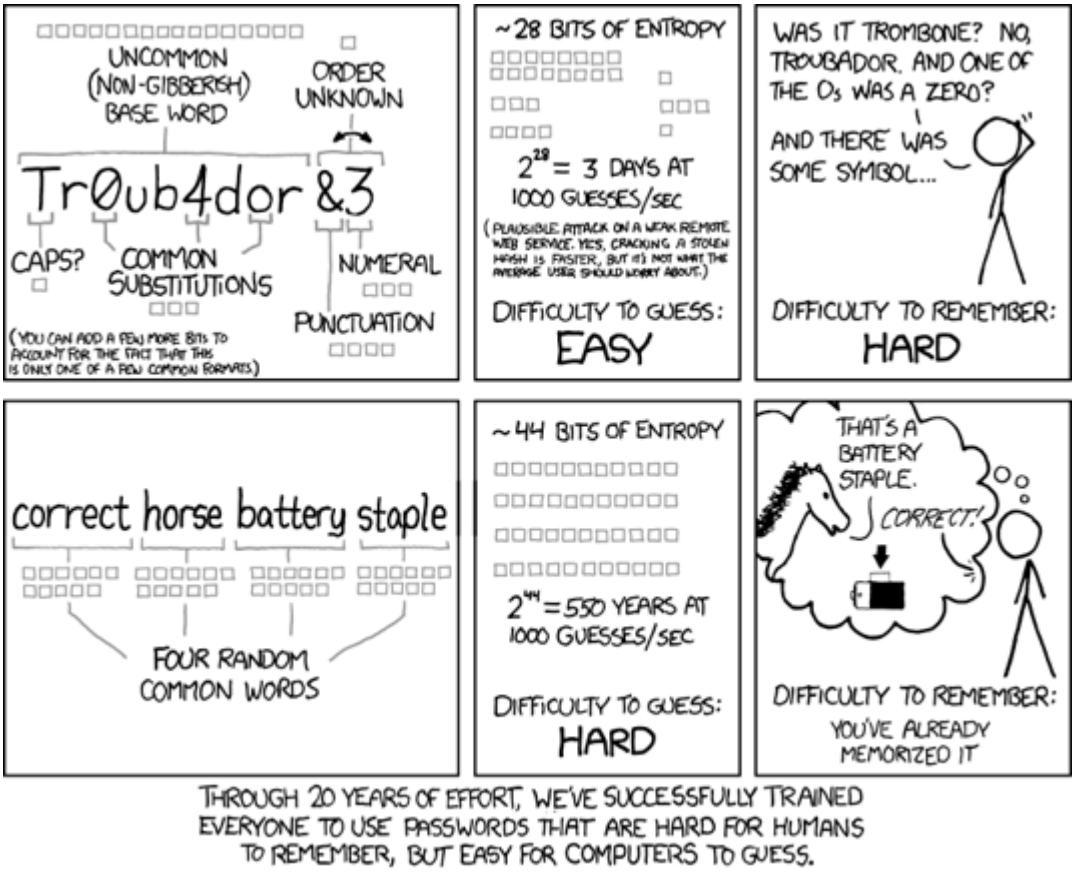
Who owns the data?

- No one knows who owns data!
  - Do you own your data?
  - Google, Facebook and others - beg to differ
- Simpler Case
  - Do you own your grades?
  - Does the school own your grade information?
  - Same argument!
- Look at sites that don't track your info
  - But - what tools do we lose?
    - Basically, is it worth the cost?
  - What is the balance?
  - With that said, your data is only as secure **as you make it**.

We are our own worst security

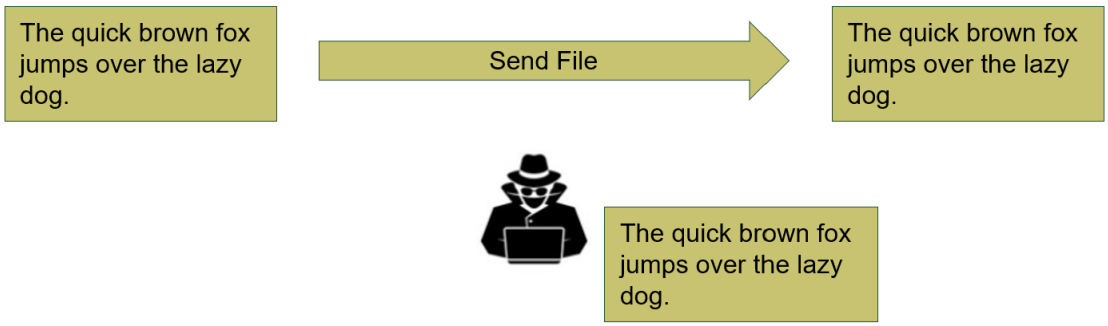
2-Step Authentication

- For signin - not encryption!
- Step 1
  - Enter your password
- Step 2
  - You get a security code (often six digits)
  - Via an app or text message
  - Enter in the code
  - OR just confirm on other devices like cell phones or security keys
    - Used in GitHub, Facebook, Google
- Really better than just a password
  - Admittedly, passwords should die
- But how to balance user experience with security?



For Next Time: Sending Data Online?

Consider the following


The quick brown fox jumps over the lazy dog.
Send File
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.