Warm-Up Activity
https://wapo.st/2HVLqUw
What is the visualization trying to show: Members of congress by decade of birth.
Why the top of the graph is not flat/curvy: number of people is discrete
What are its methods: stacked bar charts and color coding
What are the strengths / weaknesses?
Strength: intuitive and straightforward
Weakness: potentially biased color-choice

Topic 1: Evaluating Visualization Engines

Evaluating Visualization Engines

- Costs: is this engine free or do you have to pay for it?
- Functionality: Does it do what I want?
- Aesthetics: Does its plots look like plots I want to make?

Choices
Can I get ahold of this software?
Do I install it, or do I use it on a server? (ex. excel vs. google sheets)
What's the user interface like?
Is it declarative or is it imperative (procedural)? Functions of pre-built or built from ground (ex. mathematics calls); do it quickly or do it precisely

License: Software
What can you do with the software? educational/cooperate license
Can you study the software? Poorly-documented software
Who can you share it with? Local/server issue
Who can you give your visualization derivative works to?

Copyleft: share and share-alike; share-alike: if one shares your stuff, s/he have to share the stuff when s/he use them (make it publicly accessed) so to support the continual sharing.
Non-copyleft: share, but don't necessarily need to share-alike
https://choosealicense.com/
this website has a bunch of "auto" generated licences, depending on what kind of license you want your software to use
If you are ever in a position to create software, you can check this out to see what sorts of options you have.
Also, many of the open source packages we'll be using (including python) use a specific kind of license. A lot of folks use the MIT licenses, fyi.

License: Data
What can you do with the data? Be aware: Data may or may not belong to you

How do you credit that data?
Can the data be redistributed, remixed, modified?
Open data licensing / not open
http://opendefinition.org/guide/data/
https://theodi.org/guides/publishers-guide-open-data-licensing
additionally, the data might have a specific license. So maybe you can use the data for your viz, but not share the data itself with others.
How can the data be modified? Can you share the modified data?

Accessibility
Is the software installed locally on your machine?
Is it hosted at a local or remote instance? If it is on the server, to know who owns the serve is important.
Who owns the visualizations, and how is access to them controlled? licensing

Interface
How do you interact with the software?
Declarative: how do you want the plot to look? (built from imperative) make things straightforward
Imperative/Procedural: what are the steps to make the plot look that way?

Example Declarative

```
Chart(df).mark_bar().encode(
   X('precipitation', bin=True),
   Y('count(*):Q')
)
```

(From Altair Docs)

Evaluation: Costs
The "cost" of software is not exclusively the number of dollars you place on the counter when you get a big cardboard box with marketing blurbs on the side.
Think about cost in several ways:

- Monetary cost for you to use the software
- Monetary cost for someone else to view your creations: do they need download the software / do they need the licensing
- Time cost of setting up the software/system
- Cognitive cost for learning and using the system (documentation matters!)
- Transmission cost for sharing your creations

Evaluation: Aesthetics
Visualization is trendy.
When you construct something, think about the different ways it will be interpreted:

- How will the viewer understand the story of the data?

- What will the message of the visualization be? narratives
- Does the visualization say something about you and your handling of the data or utilization of tools? Ex. different industries/fields use different tools → might show professionalism

Topic 2: Markdown -> Idyll

Markdown (popular for wiki)
We'll be using markdown a bit in this course, and even more so as we move toward using Idyll.
You can think of it as standard text that we mark "up" to provide indications of formatting:

- To bold, use **two asterisks**
- You can italicize with _underscores_
- Try out making bullet lists with * on each line
- Links are fun: [link text](http://example.com/)

The lecture slides are built with Jekyll (similar to Idyll)

Idyll
The editor: https://idyll-lang.org/editor
Examples: https://idyll-lang.org/gallery
a favorite: https://idyll-lang.org/gallery/the-d-i-y-data-of-fugazi
source code: https://github.com/mathisonian/diy-data-fugazi
Install for next week/after spring break: https://idyll-lang.org/docs/getting-started

Topic 3: Maps (& more on lying with data)

Maps
A kind of visualization-making that has a long history -- left lots of problems
Thinking about map projections is important for GIS(Geographic information system) data, and generic global info viz.
Let's start by thinking about the fact that...
The Earth is a sphere.
(Fun question: to what degree is it a sphere?)
Answer: it is quite smooth. But as it spins, it flattens a little bit so it is not a perfect sphere.
Have you ever wrapped a piece of paper around a ball?

Projections (methods you use to turn the sphere into flat surfaces)
To map from one system to another, we must "project" from the original sphere to the flat object we are observing.
What are some things we could preserve during such a projection?
One common conversion from sphere to plane is the squashed cylinder approach

There's always a weird way to do it too. Here we're exploding the sphere into lots of mostly planar pieces that we can just lie out side-by-side.

Projections: Common Preservations
Typically, one or more of these will be preserved, or at least, the distortion will be minimized:

- Area
- Shape (Conformal)
- Distance

There are other properties that can be preserved, as well. Typically, maps will be a "compromise" between preserving different properties.
What happens when we preserve one property over another?

Mercator is a "conformal" projection. What is wrong with this?
conformal = shape preserving (at the expense of accurate size)
Still the one we use to navigate today (Google map)

Projections: Distortions
We can characterize distortions in a projection by examining how a known shape appears on them. The Tissot Ellipse of Distortion is a method of showing this by drawing circles of a fixed radius and examining their elliptical distortion.
so here for example, we see that the mercator projection has circles that stay circles, though they change in relative size depending on where they are on the map

Mercator: Greenland and Antarctica are HUGE
Transverse Mercator: this projection is most accurate near the vertical center line
Lambert Cylindrical: Also known as "equirectangular", this is the favorite format of NASA because it's mathematically straightforward. Note that the very top line of the image represents a single point on the globe. Pixels at the center are far lower than pixels at ends.
Mollweide: this is considered a good compromise between shape-preserving and angle preserving - but it's not perfect at either.
Sinusoidal: this has even less distortion than mollweide, but the pointy ends don't feel very elegant and planet-like
Gnomonic: this is another nightmare scenario like Mercator that was initially created for navigation. Straight lines on this map are the shortest route, but area, shape, and size are distorted.

Discussion
What happens when we make a map that minimizes one region and maximizes another?
after watching this, it's useful to know that the Peters projection is actually flawed as a teaching tool because of how much it distorts the shapes of countries near the poles.
Let's go see what Greenland actually looks like: https://thetruesize.com/
Why is Europe at the center of all the maps we've looked at? They invented the maps...

there is nothing specifically wrong with putting a pole at the center of the map
also see here that now the equator is very distorted
or why bother having a spherical or rectangular shape at all?
look how here there is very little distortion of size or shape

Maps: Coordinate Systems
Once we have our system of transformation, we need to have a method of representing positions.
Three common baseline methods:

- Spherical coordinates
- Latitude and Longitude
- Degrees, minutes, seconds

Take care with:

- Zero points
- North/South, East/West
- Ranges

Intro to cartopy
CartoPy is a toolkit that builds on matplotlib to create fast, easy map representations.
We will be relying on three key concepts:

- Axes projections (similar to our polar projections)
- Coordinate representations
- Shapes

Using these, we will be able to build out many visualizations.

CartoPy: Projections
We start out by constructing an axes in CartoPy that uses a given projection:

```
import cartopy
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111, projection=cartopy.crs.Mollweide())
ax.coastlines()
```

CartoPy: Coordinate Reference Systems
Transforming from a spherical reference system to a flat reference system is the job of the projection; transforming from one discretization of a sphere to another is the job of the coordinate system.
We can utilize Coordinate Reference Systems to describe the input coordinate system and the rasterization system are described.
For example, there are several different ways to draw "straight" lines. We can do both PlateCarree and Geodetic.

```
c_lat, c_lon = 40.1164, -88.2434
a_lat, a_lon = -18.8792, 47.5079
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection = cartopy.crs.PlateCarree())
ax.gridlines()
ax.coastlines()
ax.set_global()
ax.plot([c_lon, a_lon], [c_lat, a_lat], transform = cartopy.crs.PlateCarree())
ax.plot([c_lon, a_lon], [c_lat, a_lat], transform = cartopy.crs.Geodetic())
```

the blue line is Plate Carree, which maintains a straight line on the lat-lon grid
the orange line is Geodetic, which maintains a straight line around the curvature of the Earth
now even the blue line has some curvature because we are picking a best-of-both-worlds Mollweide projection that doesn't perfectly preserve angle or area.

Other Map Viz

- Google Maps & Earth
- WorldWide Telescope
- CesiumJS
- bqplot
- Vega & friends