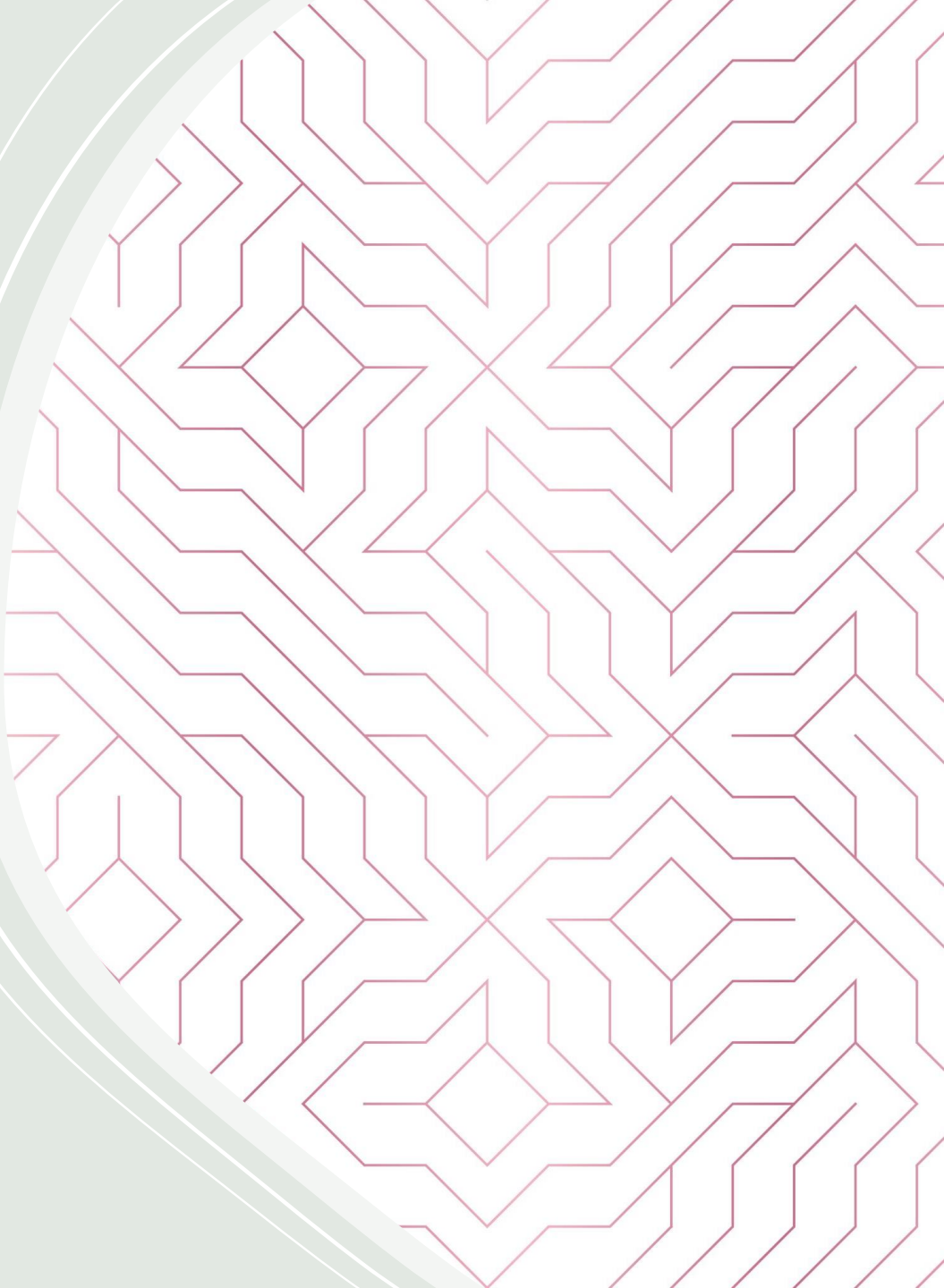
Charming the Snake: an introduction to Python

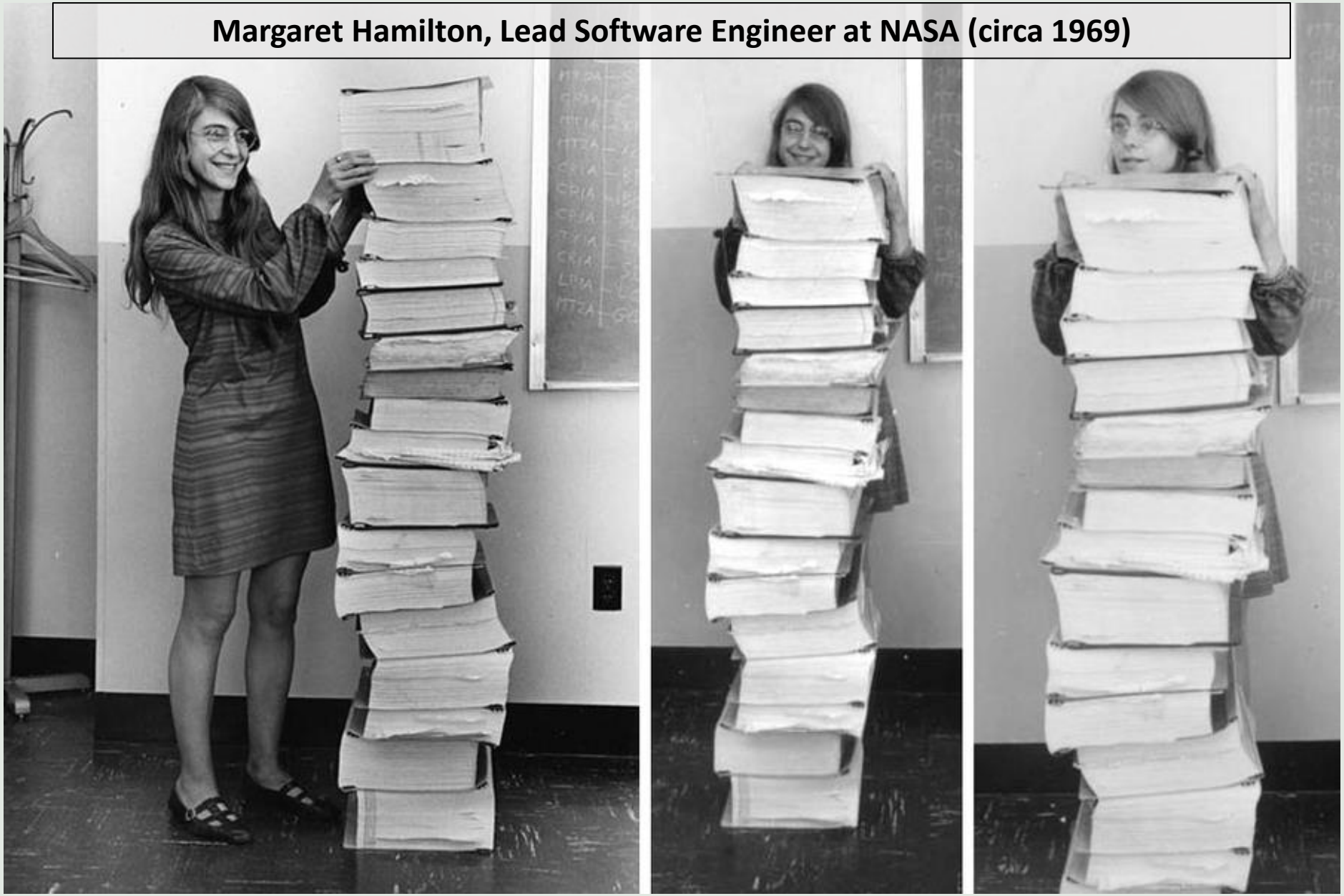


03/09/2022 – Tim Witham

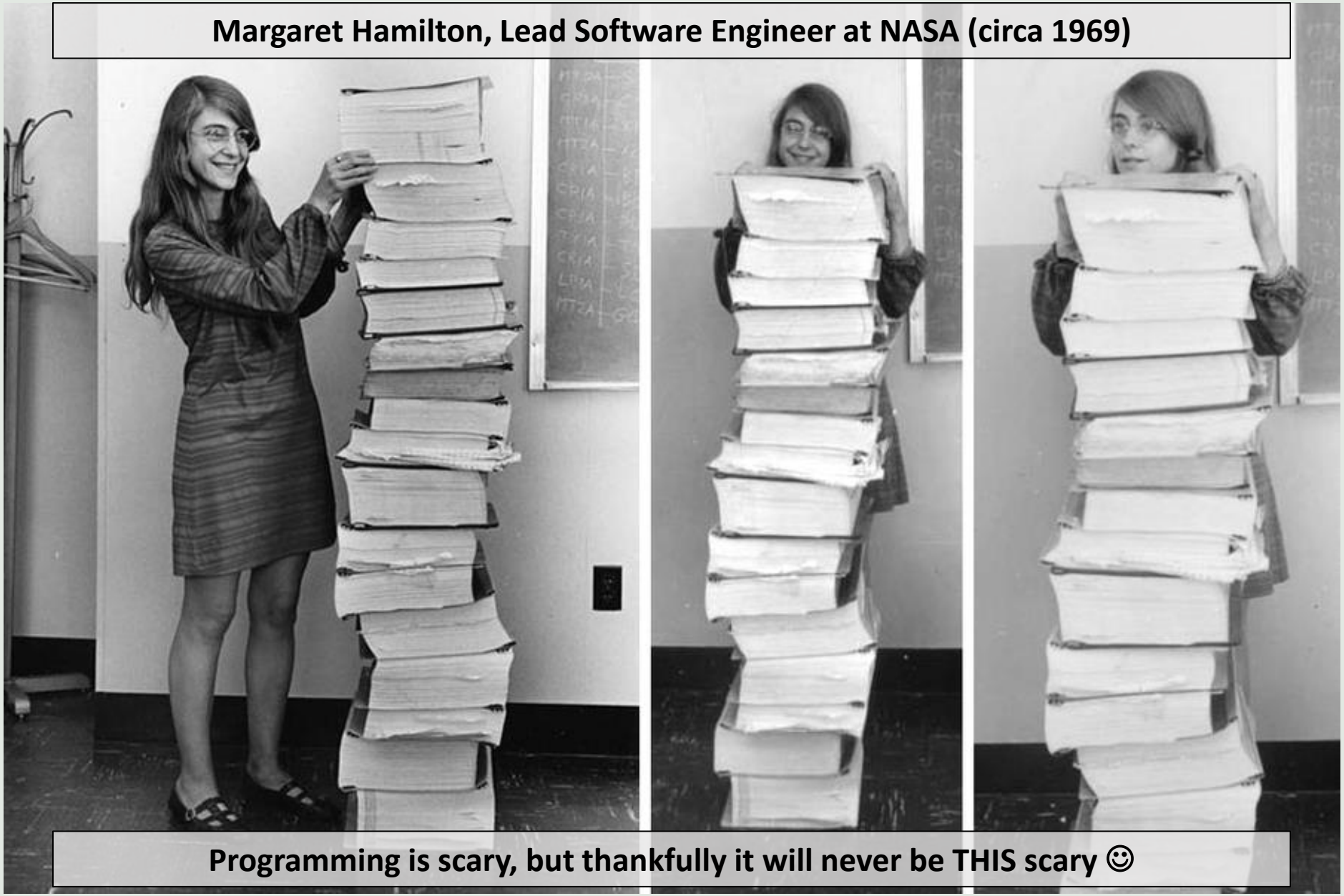
About me:

- B.S. – Geosciences – **University of Tulsa (‘17)**
  - Geoscience Student of the Year (2016) – Tulsa Geological Society
- M.S. – Geosciences – **Penn State (‘19)**
  - Imperial Barrel Award, 3rd Place (2018) – AAPG
- **Geologist at Diamondback Energy** in Midland, TX since 2019.

Margaret Hamilton, Lead Software Engineer at NASA (circa 1969)



Margaret Hamilton, Lead Software Engineer at NASA (circa 1969)



Programming is scary, but thankfully it will never be THIS scary ☺

Motivation for Learning Python

Feb 2022	Feb 2021	Change	Programming Language	Ratings	Change
1	3	▲	Python	15.33%	+4.47%
2	1	▼	C	14.08%	-2.26%
3	2	▼	Java	12.13%	+0.84%
4	4		C++	8.01%	+1.13%
5	5		C#	5.37%	+0.93%
6	6		Visual Basic	5.23%	+0.90%
7	7		JavaScript	1.83%	-0.45%
8	8		PHP	1.79%	+0.04%
9	10	▲	Assembly language	1.60%	-0.06%
10	9	▼	SQL	1.55%	-0.18%

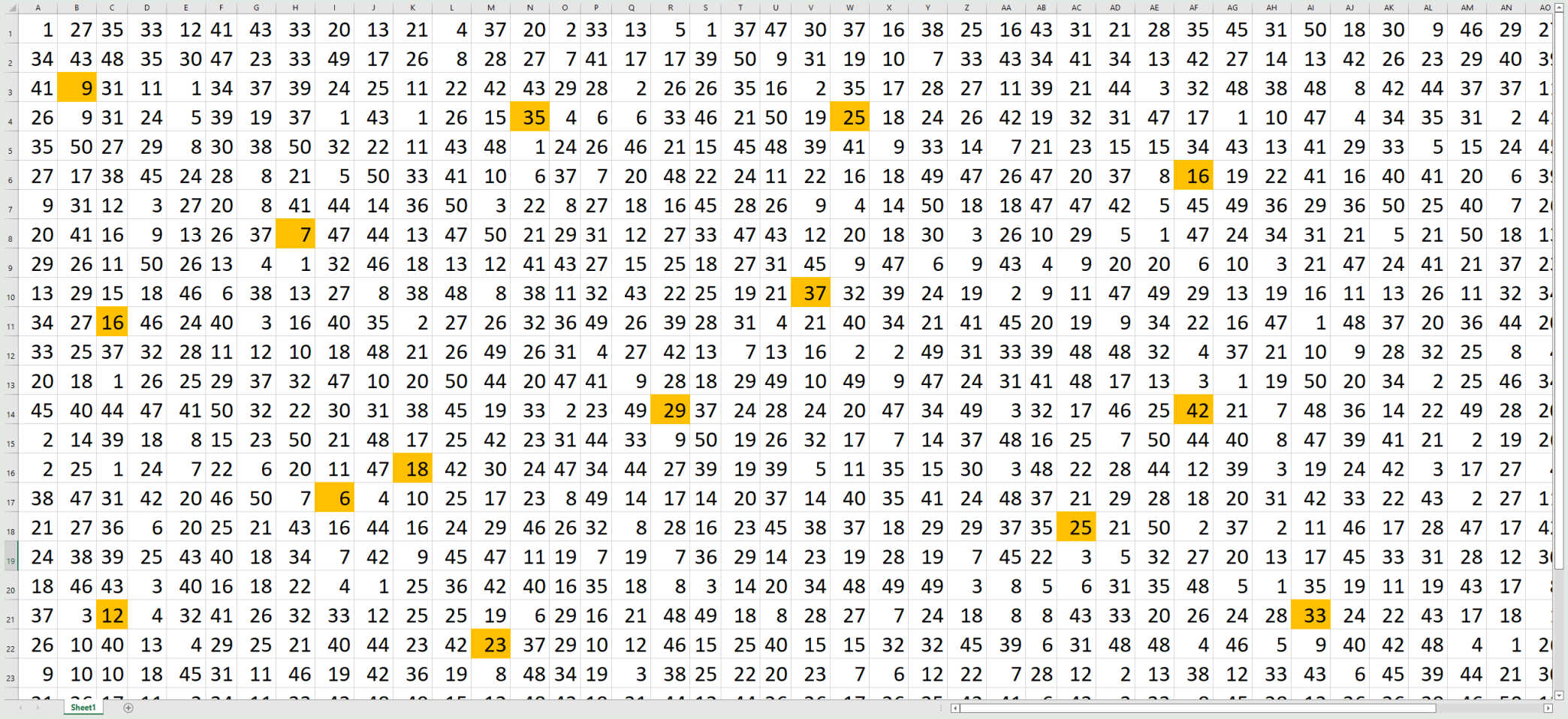
The TIOBE scale shows us which programming language is most popular!

The
Importance
Of
Being
Earnest

Oscar Wilde – 19th C. Poet

Source: ***TIOBE index*** *(February 2022)*

The Endless Spreadsheet



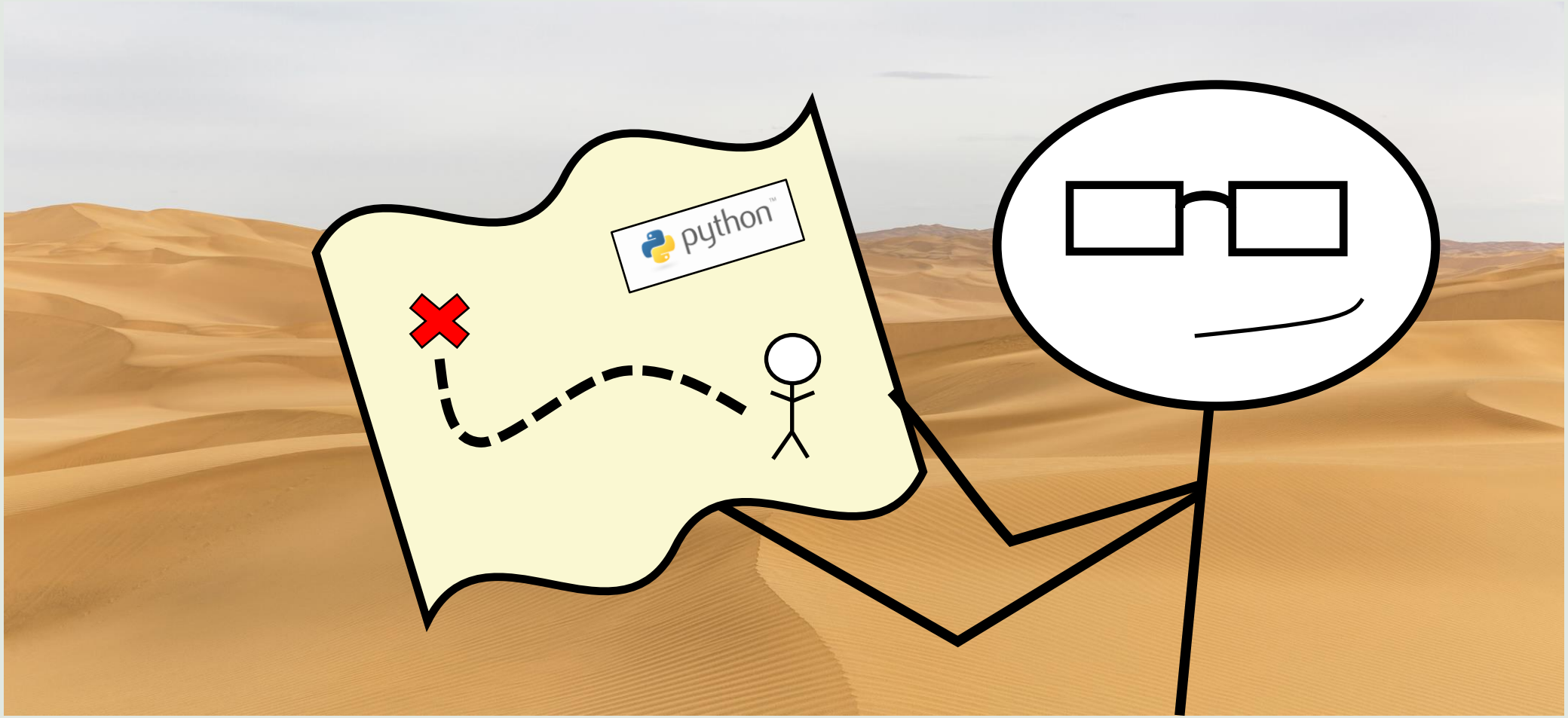
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO
1	1	27	35	33	12	41	43	33	20	13	21	4	37	20	2	33	13	5	1	37	47	30	37	16	38	25	16	43	31	21	28	35	45	31	50	18	30	9	46	29	2
2	34	43	48	35	30	47	23	33	49	17	26	8	28	27	7	41	17	17	39	50	9	31	19	10	7	33	43	34	41	34	13	42	27	14	13	42	26	23	29	40	3
3	41	9	31	11	1	34	37	39	24	25	11	22	42	43	29	28	2	26	26	35	16	2	35	17	28	27	11	39	21	44	3	32	48	38	48	8	42	44	37	37	1
4	26	9	31	24	5	39	19	37	1	43	1	26	15	35	4	6	6	33	46	21	50	19	25	18	24	26	42	19	32	31	47	17	1	10	47	4	34	35	31	2	4
5	35	50	27	29	8	30	38	50	32	22	11	43	48	1	24	26	46	21	15	45	48	39	41	9	33	14	7	21	23	15	15	34	43	13	41	29	33	5	15	24	4
6	27	17	38	45	24	28	8	21	5	50	33	41	10	6	37	7	20	48	22	24	11	22	16	18	49	47	26	47	20	37	8	16	19	22	41	16	40	41	20	6	3
7	9	31	12	3	27	20	8	41	44	14	36	50	3	22	8	27	18	16	45	28	26	9	4	14	50	18	18	47	47	42	5	45	49	36	29	36	50	25	40	7	2
8	20	41	16	9	13	26	37	7	47	44	13	47	50	21	29	31	12	27	33	47	43	12	20	18	30	3	26	10	29	5	1	47	24	34	31	21	5	21	50	18	1
9	29	26	11	50	26	13	4	1	32	46	18	13	12	41	43	27	15	25	18	27	31	45	9	47	6	9	43	4	9	20	20	6	10	3	21	47	24	41	21	37	2
10	13	29	15	18	46	6	38	13	27	8	38	48	8	38	11	32	43	22	25	19	21	37	32	39	24	19	2	9	11	47	49	29	13	19	16	11	13	26	11	32	3
11	34	27	16	46	24	40	3	16	40	35	2	27	26	32	36	49	26	39	28	31	4	21	40	34	21	41	45	20	19	9	34	22	16	47	1	48	37	20	36	44	2
12	33	25	37	32	28	11	12	10	18	48	21	26	49	26	31	4	27	42	13	7	13	16	2	2	49	31	33	39	48	48	32	4	37	21	10	9	28	32	25	8	
13	20	18	1	26	25	29	37	32	47	10	20	50	44	20	47	41	9	28	18	29	49	10	49	9	47	24	31	41	48	17	13	3	1	19	50	20	34	2	25	46	3
14	45	40	44	47	41	50	32	22	30	31	38	45	19	33	2	23	49	29	37	24	28	24	20	47	34	49	3	32	17	46	25	42	21	7	48	36	14	22	49	28	2
15	2	14	39	18	8	15	23	50	21	48	17	25	42	23	31	44	33	9	50	19	26	32	17	7	14	37	48	16	25	7	50	44	40	8	47	39	41	21	2	19	2
16	2	25	1	24	7	22	6	20	11	47	18	42	30	24	47	34	44	27	39	19	39	5	11	35	15	30	3	48	22	28	44	12	39	3	19	24	42	3	17	27	
17	38	47	31	42	20	46	50	7	6	4	10	25	17	23	8	49	14	17	14	20	37	14	40	35	41	24	48	37	21	29	28	18	20	31	42	33	22	43	2	27	1
18	21	27	36	6	20	25	21	43	16	44	16	24	29	46	26	32	8	28	16	23	45	38	37	18	29	29	37	35	25	21	50	2	37	2	11	46	17	28	47	17	4
19	24	38	39	25	43	40	18	34	7	42	9	45	47	11	19	7	19	7	36	29	14	23	19	28	19	7	45	22	3	5	32	27	20	13	17	45	33	31	28	12	3
20	18	46	43	3	40	16	18	22	4	1	25	36	42	40	16	35	18	8	3	14	20	34	48	49	49	3	8	5	6	31	35	48	5	1	35	19	11	19	43	17	
21	37	3	12	4	32	41	26	32	33	12	25	25	19	6	29	16	21	48	49	18	8	28	27	7	24	18	8	8	43	33	20	26	24	28	33	24	22	43	17	18	
22	26	10	40	13	4	29	25	21	40	44	23	42	23	37	29	10	12	46	15	25	40	15	15	32	32	45	39	6	31	48	48	4	46	5	9	40	42	48	4	1	2
23	9	10	10	18	45	31	11	46	19	42	36	19	8	48	34	19	3	38	25	22	20	23	7	6	12	22	7	28	12	2	13	38	12	33	43	6	45	39	44	21	3

Sheet1

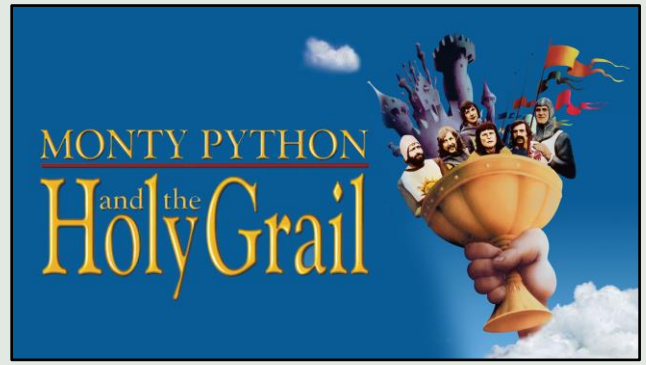
The Endless Spreadsheet

The Endless Spreadsheet

The Endless Spreadsheet



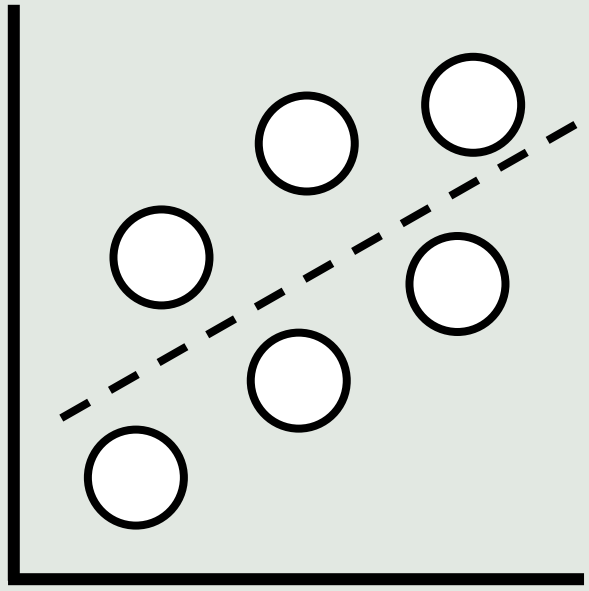
What is Python?



- It is a high level scripting language:
  - We write code down, and the computer executes it line by line.
  - “sentences” rather than big “paragraphs” of code.
- Python helps us automate the boring stuff we do every day.
- Runs tasks way faster than we ever can.
- ***IT CUTS DOWN ON ALL THE “CLICKY-CLICKY”.***
- Python is also “open-source”, meaning it is FREE!
  - Free is the best type of software according to your future boss.

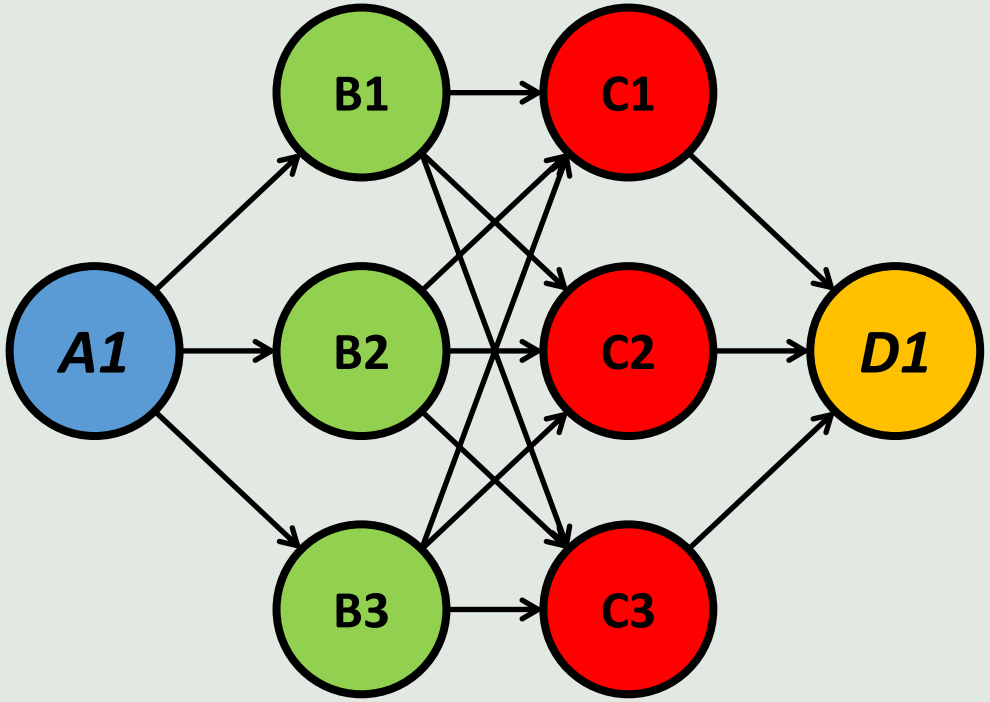
Learn the basics of data manipulation BEFORE applying complex "Machine Learning" techniques:

Simple:



Cross-plot

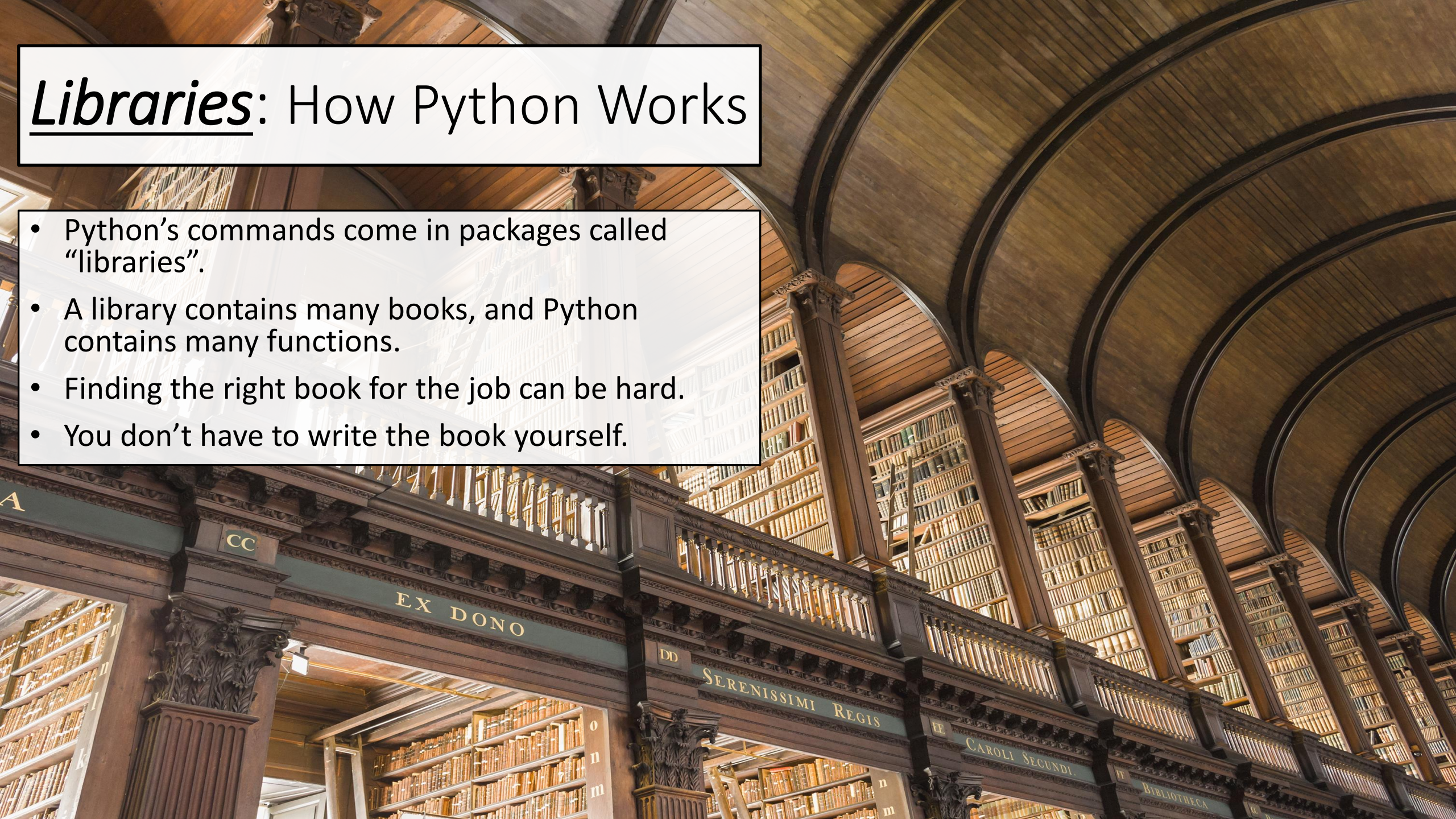
Complex:



Machine Learning

Libraries: How Python Works

- Python's commands come in packages called "libraries".
- A library contains many books, and Python contains many functions.
- Finding the right book for the job can be hard.
- You don't have to write the book yourself.



Goals of our Script:

1.) Pull data on median annual household income for the USA from Wikipedia.

2.) Plot the data up as a heat-map.

- Python will make maps faster than pointing and clicking.
- Speed is important if you will be making maps every day.

Please feel free to view my Python script and use it on your own!

https://github.com/timowith/gis_scripts/blob/main/Wiki_Web_Scrape_Py.txt

Pulling data off the internet

- Called “Web Scraping”.
- All websites have background code we can access.
- Right click on a webpage and select “inspect” to view the code.
- Use the web code to access the raw data.
- The Python library we use for this is *BeautifulSoup*.

This article is part of a series on
Income in the United States of America

Topics [hide]
Household · Personal · Affluence
Social class · Income inequality
(gender pay gap · racial pay gap)

Lists by income [show]

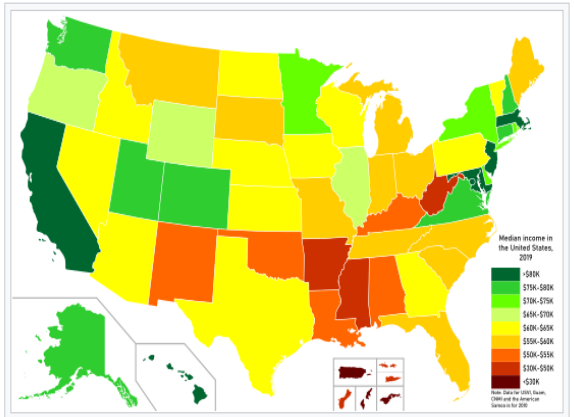
United States portal

V · T · E

States and territories ranked by median household income

Data given in current dollars. Note that tables do not reflect the margin of error in the values.[2]

State Rank	States and Washington, D.C.	2019	2018	2017	2016	2015	2014	2013	2012	2011	2010	Average annual growth rate (current dollars) in 2010-2019, %
	United States	**$65,712**	**$63,179**	**$60,336**	**$57,617**	**$55,775**	**$53,657**	**$52,250**	**$51,371**	**$50,502**	**$50,046**	**3.07%**
–	Washington, D.C.	$92,266	$85,203	$82,372	$75,506	$75,628	$71,648	$67,572	$66,583	$63,124	$60,903	4.72%
1	Maryland	$86,738	$83,242	$80,776	$78,945	$75,847	$73,971	$72,483	$71,122	$70,004	$68,854	2.6%
2	Massachusetts	$85,843	$79,835	$77,385	$75,297	$70,628	$69,160	$66,768	$65,339	$62,859	$62,072	3.67%
3	New Jersey	$85,751	$81,740	$80,088	$76,126	$72,222	$71,919	$70,165	$69,667	$67,458	$67,681	2.66%
4	Hawaii	$83,102	$80,212	$77,765	$74,511	$73,486	$69,592	$68,020	$66,259	$61,821	$63,030	3.12%
5	California	$80,440	$75,277	$71,805	$67,739	$64,500	$61,933	$60,190	$58,328	$57,287	$57,708	3.76%



Map of states by median household income in 2019

Pulling data off the internet

List of U.S. states and territories by income

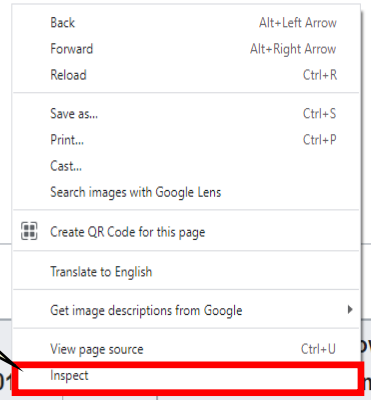
From Wikipedia, the free encyclopedia

For the list of states by income inequality, see List of U.S. states by Gini coefficient.

This is a **list of U.S. states, territories and the District of Columbia by income**. Data is given according to the 2019 American Community Survey (ACS) 1-Year Estimates, except for the American Samoa, Guam, the Northern Mariana Islands and the U.S. Virgin Islands, for which the data comes from 2010, as ACS does not operate in these areas.[note 1]

Contents [hide]

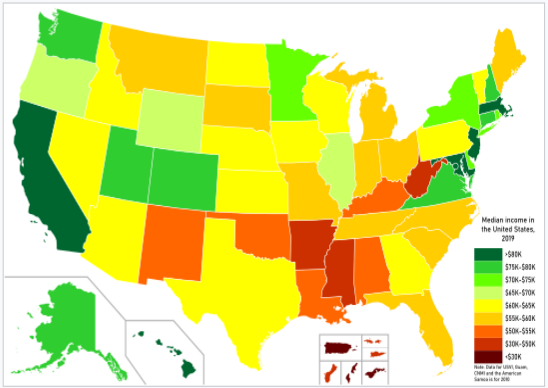
Right click on the web page and select the "Inspect" option



States and territories ranked by median household income

Data given in current dollars. Note that tables do not reflect the margin of error in the values.[2]

State Rank	States and Washington, D.C.	2019	2018	2017	2016	2015	2014	2013	2012	201		growth rate 2010-2019, %
	United States	**$65,712**	**$63,179**	**$60,336**	**$57,617**	**$55,775**	**$53,657**	**$52,250**	**$51,371**	**$50,502**	**$50,046**	**3.07%**
–	Washington, D.C.	$92,266	$85,203	$82,372	$75,506	$75,628	$71,648	$67,572	$66,583	$63,124	$60,903	4.72%
1	Maryland	$86,738	$83,242	$80,776	$78,945	$75,847	$73,971	$72,483	$71,122	$70,004	$68,854	2.6%
2	Massachusetts	$85,843	$79,835	$77,385	$75,297	$70,628	$69,160	$66,768	$65,339	$62,859	$62,072	3.67%
3	New Jersey	$85,751	$81,740	$80,088	$76,126	$72,222	$71,919	$70,165	$69,667	$67,458	$67,681	2.66%
4	Hawaii	$83,102	$80,212	$77,765	$74,511	$73,486	$69,592	$68,020	$66,259	$61,821	$63,030	3.12%
5	California	$80,440	$75,277	$71,805	$67,739	$64,500	$61,933	$60,190	$58,328	$57,287	$57,708	3.76%
6	Connecticut	$78,833	$76,348	$74,168	$73,433	$71,346	$70,048	$67,098	$67,276	$65,753	$64,032	2.34%



Map of states by median household income in 2019

This article is part of a series on

Income in the United States of America

Topics [hide]

Household · Personal · Affluence
Social class · Income inequality
(gender pay gap · racial pay gap)

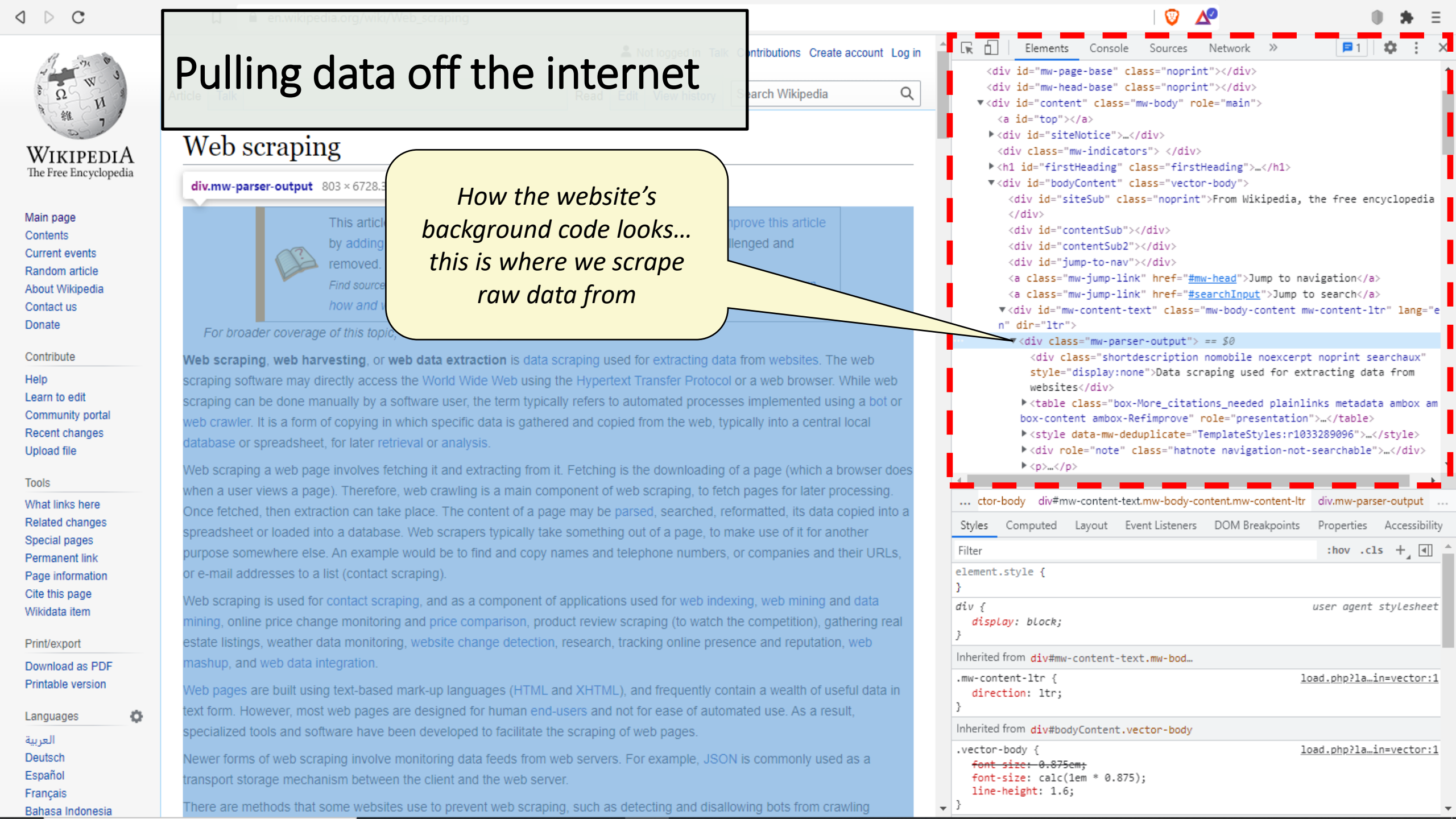
Lists by income [show]

United States portal

V · T · E

Pulling data off the internet

How the website's background code looks… this is where we scrape raw data from



Storing Data in Python

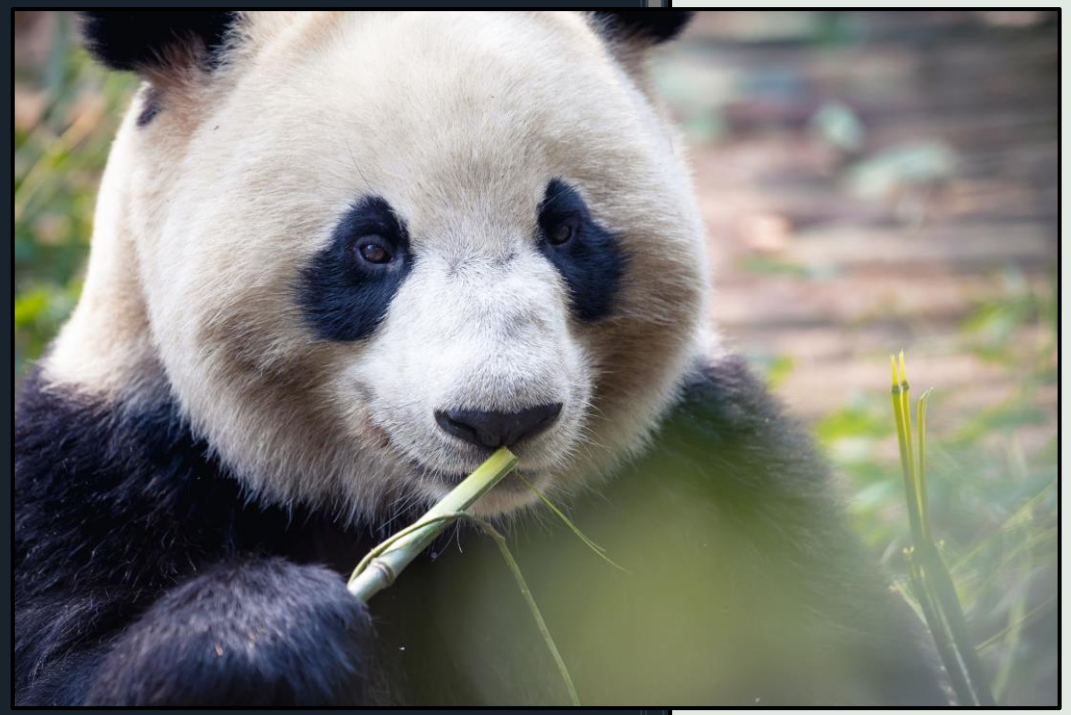
- Python can use many different data structures.
- I recommend “Dataframes”.
- Dataframes are like Excel sheets:
  - Rows and columns format.
  - We choose names for the columns.
  - Rows and columns can be selected individually.
  - Calculations can be done over the whole dataframe very fast.
- Linear Algebra is a class that teaches data structure basics.
- The primary Python library for using dataframes is called *Pandas.*

Storing Data in Python

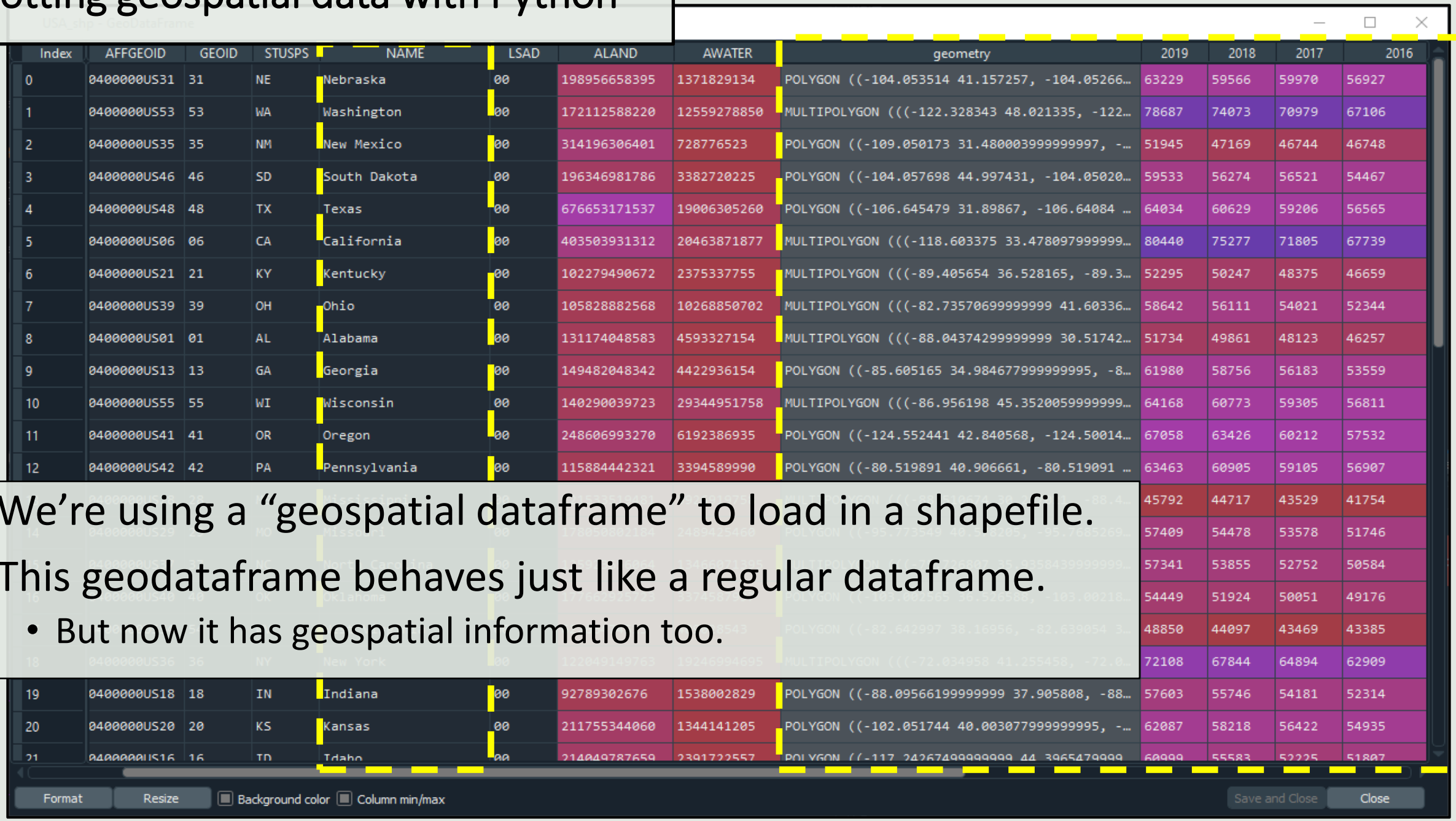
wiki_df_v2 - DataFrame

Index	NAME	2019	2018	2017	2016	2015	2014	2013	2012	2011	2010
0	United States	65712	63179	60336	57617	55775	53657	52250	51371	50502	50046
1	District of Columbia	92266	85203	82372	75506	75628	71648	67572	66583	63124	60903
2	Maryland	86738	83242	80776	78945	75847	73971	72483	71122	70004	68854
3	Massachusetts	85843	79835	77385	75297	70628	69160	66768	65339	62859	62072
4	New Jersey	85751	81740	80088	76126	72222	71919	70165	69667	67458	67681
5	Hawaii	83102	80212	77765	74511	73486	69592	68020	66259	61821	63030
6	California	80440	75277	71805	67739	64500	61933	60190	58328	57287	57708
7	Connecticut	78833	76348	74168	73433	71346	70048	67098	67276	65753	64032
8	Washington	78687	74073	70979	67106	64129	61366	58405	57573	56835	55631
9	New Hampshire	77933	74991	73381	70936	70303	66532	64230	63280	62647	61042
10	Colorado	77127	71953	69117	65685	63909	61303	58823	56765	55387	54046
11	Virginia	76456	72577	71535	68114	66262	64902	62666	61741	61882	60674
12	Utah	75780	71414	65358	65977	62912	60922	59770	57049	55869	54744
13	Alaska	75463	74346	73181	76440	73355	71583	72237	67712	67825	64576
14	Minnesota	74593	70315	68388	65599	63488	61481	60702	58906	56954	55459
15	New York	72108	67844	64894	62909	60850	58878	57369	56448	55246	54148
16	Rhode Island	71169	64340	63870	60596	58073	54891	55902	54554	53636	52254
17	Delaware	70176	64805	62852	61757	61255	59716	57846	58415	58814	55847
18	Illinois	69187	65030	62992	60960	59588	57444	56210	55137	53234	52972
19	Oregon	67058	63426	60212	57532	54148	51075	50251	49161	46816	46560
20	Wyoming	65003	61584	60434	59882	60214	57055	58752	54901	56322	53512
21	North Dakota	64577	63837	61843	60656	60557	59029	55759	53585	51704	48670

Format Resize Background color Column min/max Save and Close Close



Plotting geospatial data with Python

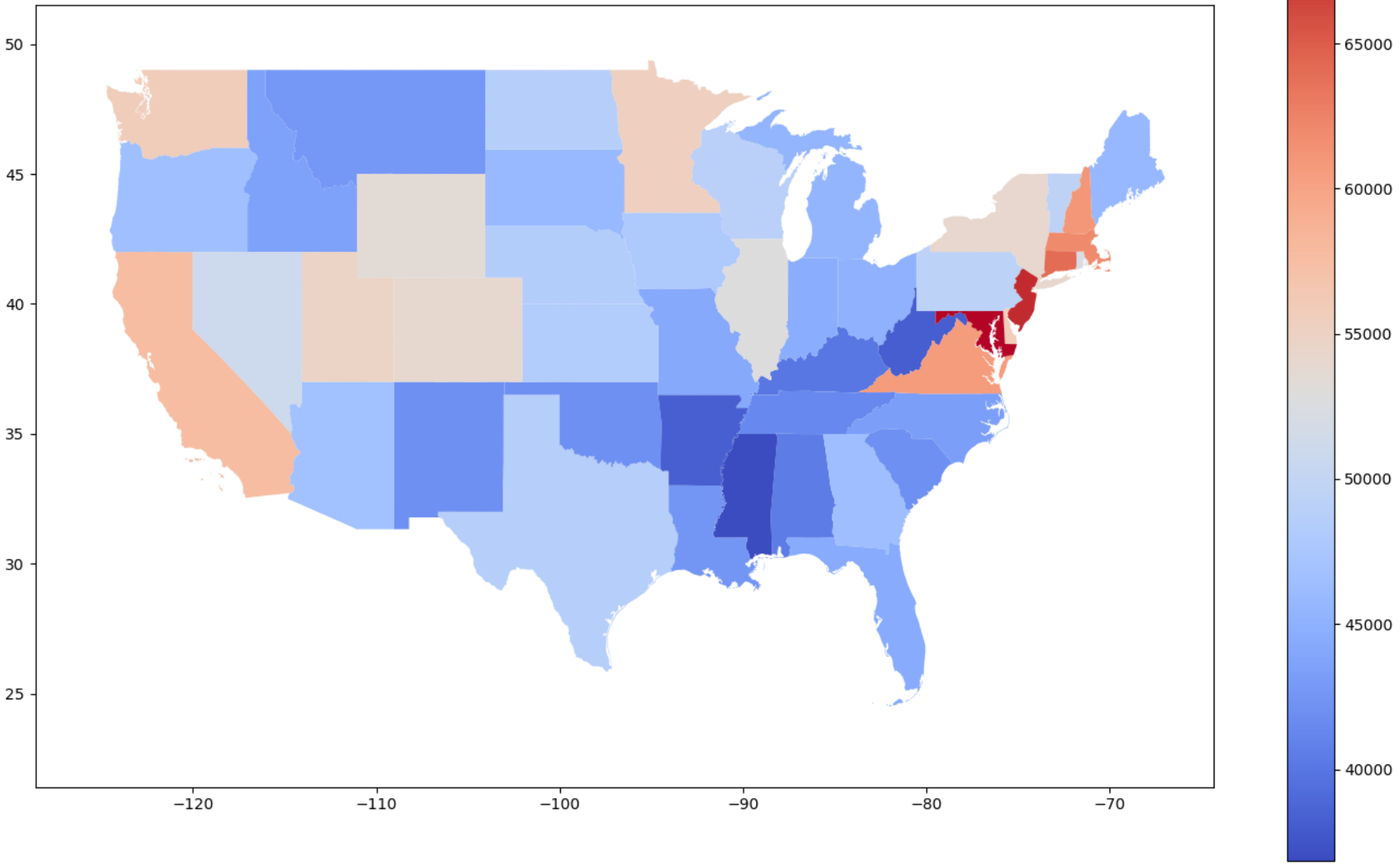


Index	AFFGEOID	GEOID	STUSPS	NAME	LSAD	ALAND	AWATER	geometry	2019	2018	2017	2016
0	0400000US31	31	NE	Nebraska	00	198956658395	1371829134	POLYGON ((-104.053514 41.157257, -104.05266…	63229	59566	59970	56927
1	0400000US53	53	WA	Washington	00	172112588220	12559278850	MULTIPOLYGON (((-122.328343 48.021335, -122…	78687	74073	70979	67106
2	0400000US35	35	NM	New Mexico	00	314196306401	728776523	POLYGON ((-109.050173 31.480003999999997, -…	51945	47169	46744	46748
3	0400000US46	46	SD	South Dakota	00	196346981786	3382720225	POLYGON ((-104.057698 44.997431, -104.05020…	59533	56274	56521	54467
4	0400000US48	48	TX	Texas	00	676653171537	19006305260	POLYGON ((-106.645479 31.89867, -106.64084 …	64034	60629	59206	56565
5	0400000US06	06	CA	California	00	403503931312	20463871877	MULTIPOLYGON (((-118.603375 33.478097999999…	80440	75277	71805	67739
6	0400000US21	21	KY	Kentucky	00	102279490672	2375337755	MULTIPOLYGON (((-89.405654 36.528165, -89.3…	52295	50247	48375	46659
7	0400000US39	39	OH	Ohio	00	105828882568	10268850702	MULTIPOLYGON (((-82.73570699999999 41.60336…	58642	56111	54021	52344
8	0400000US01	01	AL	Alabama	00	131174048583	4593327154	MULTIPOLYGON (((-88.04374299999999 30.51742…	51734	49861	48123	46257
9	0400000US13	13	GA	Georgia	00	149482048342	4422936154	POLYGON ((-85.605165 34.984677999999995, -8…	61980	58756	56183	53559
10	0400000US55	55	WI	Wisconsin	00	140290039723	29344951758	MULTIPOLYGON (((-86.956198 45.3520059999999…	64168	60773	59305	56811
11	0400000US41	41	OR	Oregon	00	248606993270	6192386935	POLYGON ((-124.552441 42.840568, -124.50014…	67058	63426	60212	57532
12	0400000US42	42	PA	Pennsylvania	00	115884442321	3394589990	POLYGON ((-80.519891 40.906661, -80.519091 …	63463	60905	59105	56907
									45792	44717	43529	41754
									57409	54478	53578	51746
									57341	53855	52752	50584
									54449	51924	50051	49176
									48850	44097	43469	43385
									72108	67844	64894	62909
19	0400000US18	18	IN	Indiana	00	92789302676	1538002829	POLYGON ((-88.09566199999999 37.905808, -88…	57603	55746	54181	52314
20	0400000US20	20	KS	Kansas	00	211755344060	1344141205	POLYGON ((-102.051744 40.003077999999995, -…	62087	58218	56422	54935
21	0400000US16	16	ID	Idaho	00	214049787659	2391722557	POLYGON ((-117.24267499999999 44.3965479999…	60999	55583	52225	51807

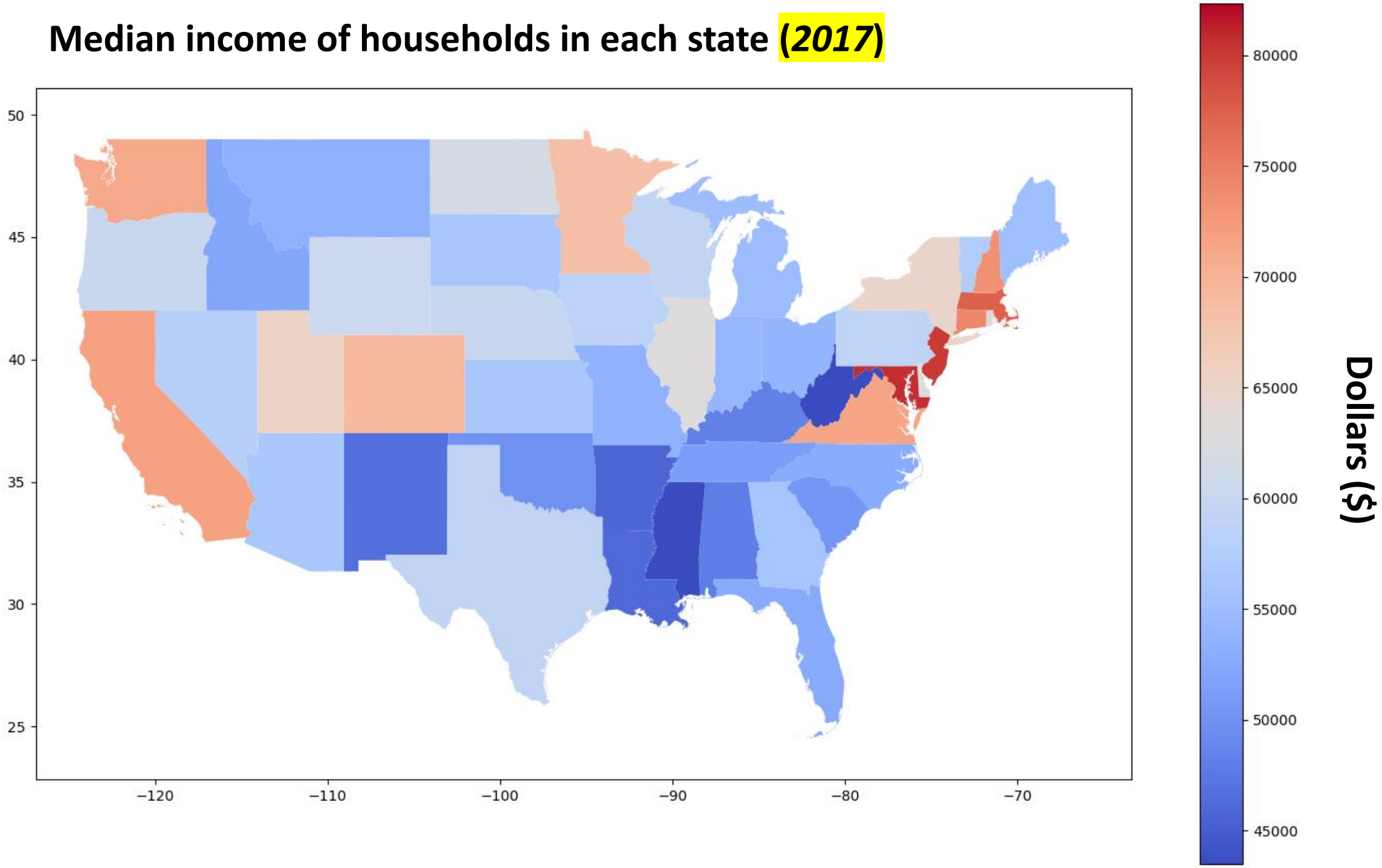
Format | Resize | Background color | Column min/max | Save and Close | Close

- We're using a "geospatial dataframe" to load in a shapefile.
- This geodataframe behaves just like a regular dataframe.
  - But now it has geospatial information too.

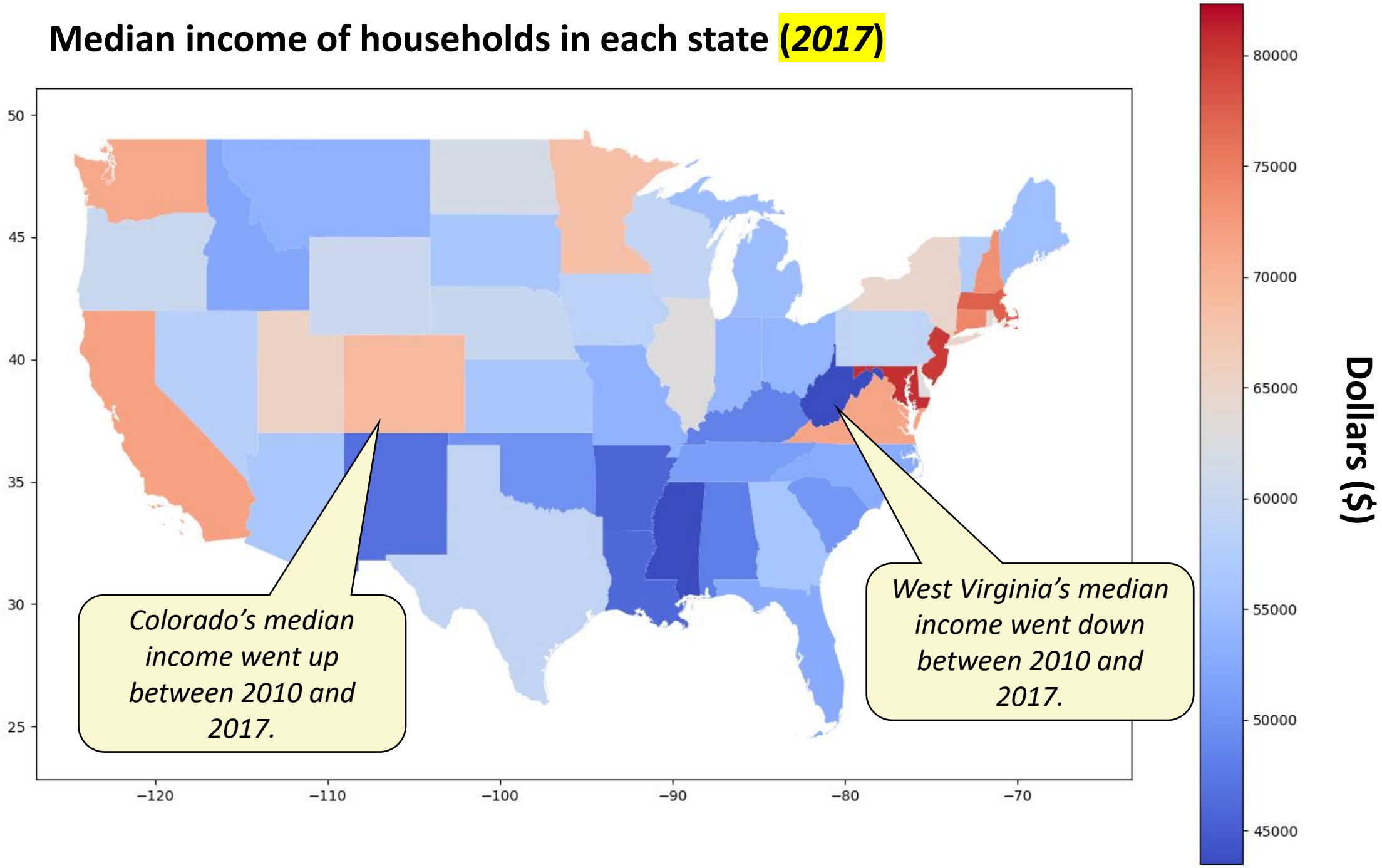
Median income of households in each state (*2010*)



Median income of households in each state (*2017*)



Median income of households in each state (*2017*)



Conclusion

- Programming skills will make you stand out.
- Choose one of the top 5 most popular languages to learn.
  - Python!
- DataFrames are a very concise way to store your data.

Sources

- https://medium.com/analytics-vidhya/web-scraping-a-wikipedia-table-into-a-dataframe-c52617e1f451
- https://geopandas.org/en/stable/getting_started/install.html
- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
- https://www.tiobe.com/tiobe-index/

Important Python Libraries

- Data Organization:
  - pandas
- Geographic Mapping:
  - geopandas
- Manipulating and searching through strings:
  - re (short for ‘regex’)
- General mathematics:
  - Numpy (short for ‘Numeric Python’)
- General scientific concepts:
  - scipy (short for ‘Scientific Python’)
- Web scraping:
  - BeautifulSoup
- Pathway and folder manipulation:
  - os
  - Glob
- Plotting/graphs:
  - matplotlib
- Machine Learning:
  - TensorFlow
  - Keras
  - Scikit-learn

Additional programming resources

Python is a great place to start for learning programming. It's intuitive and open source, meaning that anyone can use it for free. This also means that a lot of people around the are constantly adding new features to it. According to the TIOBE index, which measures how popular different programming languages are, Python is the #3 most used programming language worldwide as of March 2021 (https://www.tiobe.com/tiobe-index/).

For installation, I would recommend downloading "Anaconda", it's a one stop shop program that contains Python and few other programming languages. Once you download "Anaconda", I would recommend coding Python within "Spyder" or "Jupyter", both are Python IDEs included within Anaconda.
Keep in mind that the current release of Python is 3.9.2. So make sure any tutorials you utilize are teaching to Python 3 when possible.
Link to Anaconda download: https://www.anaconda.com/products/individual

Introduction:

- CodeAcademy: This is where I started when I was learning programming. When I used it last, you could perform programming exercises entirely from within your web browser, so you didn't need to download anything. Very useful and quick to get into. Link: https://www.codecademy.com/learn/learn-python-3
- Khan Academy: a multifaceted website with a bunch of different tutorials for a wide range of science and math topics. A lot of their videos are on YouTube. Here's a Python tutorial on YouTube: https://www.youtube.com/watch?v=husPzLE6sZc&list=PLJR1V_NHIKrCkswPMULzQFHpYa57ZFGbs.

General Classes/Certificates:

The Code Academy link from earlier in this email has a certificate.

- Coursera.org has a lot of good professional certificates. I've participated in a few of their classes. Depending on the professor teaching the class, the quality will vary.
  https://www.coursera.org/specializations/python?utm_source=gg&utm_medium=sem&utm_campaign=06-PythonforEverybody-US&utm_content=06-PythonforEverybody-US&campaignid=300366907&adgroupid=34186056677&device=c&keyword=coursera%20python&matchtype=e&network=g&devicemodel=&adpostion=&creativeid=273185019772&hide_mobile_promo&gclid=EAIaIQobChMIzrDy8a_M7wIVg5uGCh1b6QnnEAAYASAAEgKlQfD_BwE
- General blog post describing the different Python certificates available:
  https://www.dataquest.io/blog/python-certification/
  They make a good general point that a certificate is only as good as the portfolio of programs you have written. I personally think a certificate is nice, as long as you also have a list of projects you've worked on to supplement the certificate.

Reference Materials:

- ***StackOverflow***: An incredibly helpful website. It's like the Google of programming questions. Whenever I don't know how to do something in Python, I go to this website and type in my question. Other users might've had the same question and other programmers provide their answers and insights for free. Here's an example of what a question and answer on this website looks like (https://stackoverflow.com/questions/27885397/how-do-i-install-a-python-package-with-a-whl-file/27909082#27909082). Just type in the search bar at the top if you have a question.

- List of good libraries to use within Python:
  https://www.dataquest.io/blog/15-python-libraries-for-data-science/

Raw code

```
# 02/28/2022: Writing a webscrape script that takes relevant data out of a website and populates a dataframe

# From there, use GeoPandas to plot the scraped data up as a choropleth map.



# Checking some background environment stuff:

import sys

sys.executable

import platform

platform.architecture()



# Import most critical libraries for this script:



# Import HTML libraries:

import requests

from bs4 import BeautifulSoup

# Import REGEX library if needed (can stay commented out):

# import re



# Import dataframe/geospatial dataframe libraries:

import pandas as pd

import geopandas as gpd

from shapely.geometry import Point, Polygon
```

```
# Import one pure plotting library (will help with making nice looking legends at the end of this script):

import matplotlib.pyplot as plt



# IMPORTANT NOTE to future users:

# I had a LOT of trouble getting the 'geopandas' library installed properly using pip... I would recommend downloading the raw "wheel" files for

# the prerequisite "GDAL" and "Fiona" packages from these links (download only the package that matches your system's architecture (32 or 64 bit) and the corresponding Python version):

    # https://www.lfd.uci.edu/~gohlke/pythonlibs/#gdal

    # https://www.lfd.uci.edu/~gohlke/pythonlibs/#fiona

# Once these are downloaded, type the following in the command line below in Spyder or your Python IDE of choice (MAKE SURE GDAL IS INSTALLED BEFORE FIONA!):

    # pip install path/to/downloaded/GDALfile.whl

    # pip install path/to/downloaded/fionafile.whl

# Once those are installed, then type the following in the command line and you should be good to go:

    # pip install geopandas
```

```
# Refer to this GIS StackExchange discussion if you still need more help:

    # https://gis.stackexchange.com/questions/330840/error-installing-geopandas



# # # --- PART 1: LOAD THE TABLE FROM THE WEB --- # # #



# Ping the URL we want to extract a wiki table from:

URL = "https://en.wikipedia.org/wiki/List_of_U.S._states_and_territories_by_income"



# random testing next two lines:

# page = requests.get(URL)

# table_class = "wikitable sortable jquery-tablesorter"



# Save the HTML response:

response = requests.get(URL)



# Parse the above raw html table/data response into a BeautifulSoup object (a cleaner way to look at the response):

soupy = BeautifulSoup(response.text,'html.parser')

# find the sub-object in that HTML response that is of class 'wikitable':

usa_table = soupy.find('table', {'class':"wikitable"})
```

```
# Note to self, what if there were multiple wikitables per url? Would we have to iterate then? Probably... worth looking into later.



# Pandas has some built in functions that can read HTML, use these to make our DF:

wiki_df = pd.read_html(str(usa_table))

# convert list to dataframe below (because dataframes have more functionality than lists):

wiki_df = pd.DataFrame(wiki_df[0])



# # # --- PART 2: REFORMAT THE DATAFRAME SLIGHTLY --- # # #



# Looking at our fresh and brand new dataframe, we can see there might be some issues with plotting the data on a map.

# First of all, there are some columns we are not interested in for the purposes of this script.

# Namely, we don't care about the 'State Rank' and 'Growth Rate' columns, so let's drop them:

wiki_df_v2 = wiki_df.drop(columns = ['State Rank','Average annual growth rate (current dollars) in 2010-2019,\xa0%'], axis = 1)



# Cool, now that we've narrowed our data frame down to the columns we want, let's get rid of the dollar signs and convert all the numbers from str to int.

# We convert from string (str) to integer (int) because a lot of the plotting functions in Python require numerical values to be actual integers or float (decimal) numbers.

# If we try to plot a string, even though that string might be made up of numerical characters, we will get errors.

# Said another way: unless the numerals are actual numbers, we will see errors returned by our script whenever we try to run it.

# We're going to loop through a list of only the column names we want to remove dollar signs and commas in the numbers from.

# Make a list of the matching column names, this will be used for column selection in the

target_column_list = list(wiki_df_v2.columns)



# Now, loop through the first through last items in the target_column_list :

for i in list(range(1,(len(target_column_list)),1)):

    wiki_df_v2[target_column_list[i]] = wiki_df_v2[target_column_list[i]].str.replace(',', '')

    wiki_df_v2[target_column_list[i]] = wiki_df_v2[target_column_list[i]].str.replace(r'$', '')

    wiki_df_v2[target_column_list[i]] = wiki_df_v2[target_column_list[i]].astype(int)
```

```
# # # --- PART 3: GEOPANDAS TIME --- # # #



# You will need to download a shapefile of state polygons, I used the US census website below to find a good one:

# https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html

USA_shp = gpd.read_file('G:\TWitham\DataRole\Testing_Python\cb_2018_us_state_5m.shp')

# Note that the names of Washington D.C. differ between this shapefile we just loaded in and

# the equivalent location within the wiki_df_v2. Rename "Washington DC" to "District of Columbia"

wiki_df_v2.at[1,'States and Washington, D.C.'] = 'District of Columbia'

# ^ Little data cleanups like this are often necessary to do when getting data ready for actual analysis.

# We also want to rename the "States" column in wiki_df_v2 to "NAME" so the left join with USA_shp will be as seamless as possible:

wiki_df_v2 = wiki_df_v2.rename(columns = {'States and Washington, D.C.' : 'NAME'})



# Now, we are going to do a merge (specifically, a left join) on the two data sets: the geodataframe named "USA_shp" & the regular Pandas dataframe named "wiki_df_v2":

# NOTE! The Geodataframe should always be the left side of the join, otherwise we will remove the geometry information which is most crucial for Python to be able

# to draw our map automatically for us.

# if you would like to read further about attribute joins in geopandas, I recommend reading their documentation here: https://geopandas.org/en/stable/docs/user_guide/mergingdata.html

USA_shp = USA_shp.merge(wiki_df_v2, on = 'NAME')

# Great, the merge above worked perfectly! We joined the geodataframe (USA_shp) to the regular Wikipedia dataframe (wiki_df_v2).

# All that's left now is to make a choropleth map of each state, colored by the average household income.

# Make a copy of USA_shp with a shorter name (will make for easier to read code when plotting):

USA = USA_shp

# pick a year of your choice from this list: 2019,2018,2017,2016,2015,2014,2013,2012,2011, or 2010.

# I graduated from TU in 2017, so let's pick that year.

year_of_choice = '2017'

# !!! Now, Make a map plot of our data !!! #

# default, basic plot is below, but let's make it prettier.

# USA.plot(column = year_of_choice);

# Start by adding a legend using matplotlib:

# Add a legend:

fig, ax = plt.subplots(1,1)

USA.plot(column = year_of_choice, ax = ax, legend = True)

# !!! NOTE that the map will appear in a separate window down in your taskbar. Click on the icon in your taskbar to view the map.

# Make sure to zoom in on the map using the "magnifying glass" button on the top left if you want to zoom in on the continental USA.
```

Raw code

```
10
11  # Checking some background environment stuff:
12  import sys
13  sys.executable
14  import platform
15  platform.architecture()
16
17  # Import most critical libraries for this script:
18
19  # Import HTML libraries:
20  import requests
21  from bs4 import BeautifulSoup
22  # Import REGEX library if needed (can stay commented out):
23  # import re
24
25  # Import dataframe/geospatial dataframe libraries:
26  import pandas as pd
27  import geopandas as gpd
28  from shapely.geometry import Point, Polygon
29
30  # Import one pure plotting library (will help with making nice looking legends at the end of this script):
31  import matplotlib.pyplot as plt
32
33  # IMPORTANT NOTE to future users:
34  # I had a LOT of trouble getting the 'geopandas' library installed properly using pip... I would recommend downloading the raw "wheel" files for
35  # the prerequisite "GDAL" and "Fiona" packages from these links (download only the package that matches your system's architecture (32 or 64 bit) and the corresponding Python version):
36      # https://www.lfd.uci.edu/~gohlke/pythonlibs/#gdal
37      # https://www.lfd.uci.edu/~gohlke/pythonlibs/#fiona
38
39  # Once these are downloaded, type the following in the command line below in Spyder or your Python IDE of choice (MAKE SURE GDAL IS INSTALLED BEFORE FIONA!):
40      # pip install path/to/downloaded/GDALfile.whl
41      # pip install path/to/downloaded/fionafile.whl
42
43  # Once those are installed, then type the following in the command line and you should be good to go:
44      # pip install geopandas
45
46  # Refer to this GIS StackExchange discussion if you still need more help:
47      # https://gis.stackexchange.com/questions/330840/error-installing-geopandas
```

Raw code

```
47      # https://gis.stackexchange.com/questions/330840/error-installing-geopandas
48
49  # # # --- PART 1: LOAD THE TABLE FROM THE WEB --- # # #
50
51  # Ping the URL we want to extract a wiki table from:
52  URL = "https://en.wikipedia.org/wiki/List_of_U.S._states_and_territories_by_income"
53
54  # random testing next two lines:
55  # page = requests.get(URL)
56  # table_class = "wikitable sortable jquery-tablesorter"
57
58  # Save the HTML response:
59  response = requests.get(URL)
60
61  # Parse the above raw html table/data response into a BeautifulSoup object (a cleaner way to look at the response):
62  soupy = BeautifulSoup(response.text,'html.parser')
63  # find the sub-object in that HTML response that is of class 'wikitable':
64  usa_table = soupy.find('table', {'class':"wikitable"})
65  # Note to self, what if there were multiple wikitables per url? Would we have to iterate then? Probably... worth looking into later.
66
67  # Pandas has some built in functions that can read HTML, use these to make our DF:
68  wiki_df = pd.read_html(str(usa_table))
69  # convert list to dataframe below (because dataframes have more functionality than lists):
70  wiki_df = pd.DataFrame(wiki_df[0])
71
72  # # # --- PART 2: REFORMAT THE DATAFRAME SLIGHTLY --- # # #
73
74  # Looking at our fresh and brand new dataframe, we can see there might be some issues with plotting the data on a map.
75  # First of all, there are some columns we are not interested in for the purposes of this script.
76  # Namely, we don't care about the 'State Rank' and 'Growth Rate' columns, so let's drop them:
77
78  wiki_df_v2 = wiki_df.drop(columns = ['State Rank','Average annual growth rate (current dollars) in 2010-2019,\xa0%'], axis = 1)
79
80  # Cool, now that we've narrowed our data frame down to the columns we want, let's get rid of the dollar signs and convert all the numbers from str to int.
81  # We convert from string (str) to integer (int) because a lot of the plotting functions in Python require numerical values to be actual integers or float (decimal) numbers.
82  # If we try to plot a string, even though that string might be made up of numerical characters, we will get errors.
83  # Said another way: unless the numerals are actual numbers, we will see errors returned by our script whenever we try to run it.
84
85  # We're going to loop through a list of only the column names we want to remove dollar signs and commas in the numbers from.
86  # Make a list of the matching column names, this will be used for column selection in the
87  target_column_list = list(wiki_df_v2.columns)
88
```

Raw code

```
88
89  # Now, loop through the first through last items in the target_column_list :
90  for i in list(range(1,(len(target_column_list)),1)):
91      wiki_df_v2[target_column_list[i]] = wiki_df_v2[target_column_list[i]].str.replace(',', '')
92      wiki_df_v2[target_column_list[i]] = wiki_df_v2[target_column_list[i]].str.replace(r'$', '')
93      wiki_df_v2[target_column_list[i]] = wiki_df_v2[target_column_list[i]].astype(int)
94
95
96
97  # # # --- PART 3: GEOPANDAS TIME --- # # #
98
99  # You will need to download a shapefile of state polygons, I used the US census website below to find a good one:
100 # https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html
101
102
103 USA_shp = gpd.read_file('G:\TWitham\DataRole\Testing_Python\cb_2018_us_state_5m.shp')
104
105 # Note that the names of Washington D.C. differ between this shapefile we just loaded in and
106 # the equivalent location within the wiki_df_v2. Rename "Washington DC" to "District of Columbia"
107
108 wiki_df_v2.at[1,'States and Washington, D.C.'] = 'District of Columbia'
109 # ^ Little data cleanups like this are often necessary to do when getting data ready for actual analysis.
110
111 # We also want to rename the "States" column in wiki_df_v2 to "NAME" so the left join with USA_shp will be as seamless as possible:
112 wiki_df_v2 = wiki_df_v2.rename(columns = {'States and Washington, D.C.' : 'NAME'})
113
114 # Now, we are going to do a merge (specifically, a left join) on the two data sets: the geodataframe named "USA_shp" & the regular Pandas dataframe named "wiki_df_v2":
115 # NOTE! The Geodataframe should always be the left side of the join, otherwise we will remove the geometry information which is most crucial for Python to be able
116 # to draw our map automatically for us.
117
118 # if you would like to read further about attribute joins in geopandas, I recommend reading their documentation here: https://geopandas.org/en/stable/docs/user_guide/mergingdata.html
119
120 USA_shp = USA_shp.merge(wiki_df_v2, on = 'NAME')
121
122 # Great, the merge above worked perfectly! We joined the geodataframe (USA_shp) to the regular Wikipedia dataframe (wiki_df_v2).
123
124 # All that's left now is to make a choropleth map of each state, colored by the average household income.
125
126 # Make a copy of USA_shp with a shorter name (will make for easier to read code when plotting):
127 USA = USA_shp
```

Raw code

```
106    # the equivalent location within the wiki_df_v2. Rename "Washington DC" to "District of Columbia"
107
108    wiki_df_v2.at[1,'States and Washington, D.C.'] = 'District of Columbia'
109    # ^ Little data cleanups like this are often necessary to do when getting data ready for actual analysis.
110
111    # We also want to rename the "States" column in wiki_df_v2 to "NAME" so the left join with USA_shp will be as seamless as possible:
112    wiki_df_v2 = wiki_df_v2.rename(columns = {'States and Washington, D.C.' : 'NAME'})
113
114    # Now, we are going to do a merge (specifically, a left join) on the two data sets: the geodataframe named "USA_shp" & the regular Pandas dataframe named "wiki_df_v2":
115    # NOTE! The Geodataframe should always be the left side of the join, otherwise we will remove the geometry information which is most crucial for Python to be able
116    # to draw our map automatically for us.
117
118    # if you would like to read further about attribute joins in geopandas, I recommend reading their documentation here: https://geopandas.org/en/stable/docs/user_guide/mergingdata.html
119
120    USA_shp = USA_shp.merge(wiki_df_v2, on = 'NAME')
121
122    # Great, the merge above worked perfectly! We joined the geodataframe (USA_shp) to the regular Wikipedia dataframe (wiki_df_v2).
123
124    # All that's left now is to make a choropleth map of each state, colored by the average household income.
125
126    # Make a copy of USA_shp with a shorter name (will make for easier to read code when plotting):
127    USA = USA_shp
128
129    # pick a year of your choice from this list: 2019,2018,2017,2016,2015,2014,2013,2012,2011, or 2010.
130    # I graduated from TU in 2017, so let's pick that year.
131    year_of_choice = '2017'
132
133    # !!! Now, Make a map plot of our data !!! #
134
135    # default, basic plot is below, but let's make it prettier.
136    # USA.plot(column = year_of_choice);
137
138    # Start by adding a legend using matplotlib:
139    # Add a legend:
140    fig, ax = plt.subplots(1,1)
141    USA.plot(column = year_of_choice, ax = ax, legend = True)
142    # !!! NOTE that the map will appear in a separate window down in your taskbar. Click on the icon in your taskbar to view the map.
143    # Make sure to zoom in on the map using the "magnifying glass" button on the top left if you want to zoom in on the continental USA.
144
145
146
147
```