Lecture 4d
Google Earth Engine

Pierre Biscaye

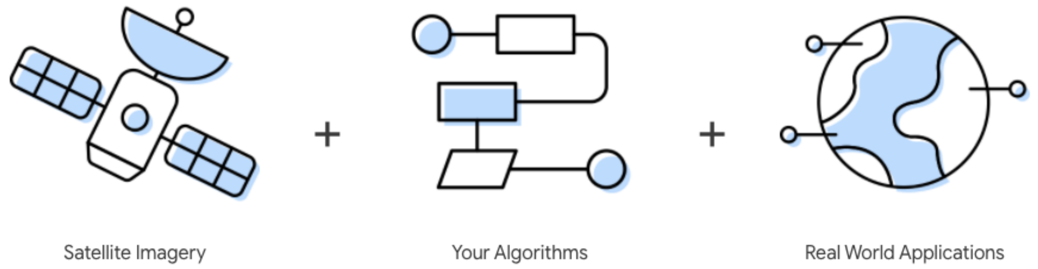
Université Clermont Auvergne

Data Science for Economics

Note: Materials for this lecture are drawn from Josh Blumenstock's Big Data & Development course at UC Berkeley.

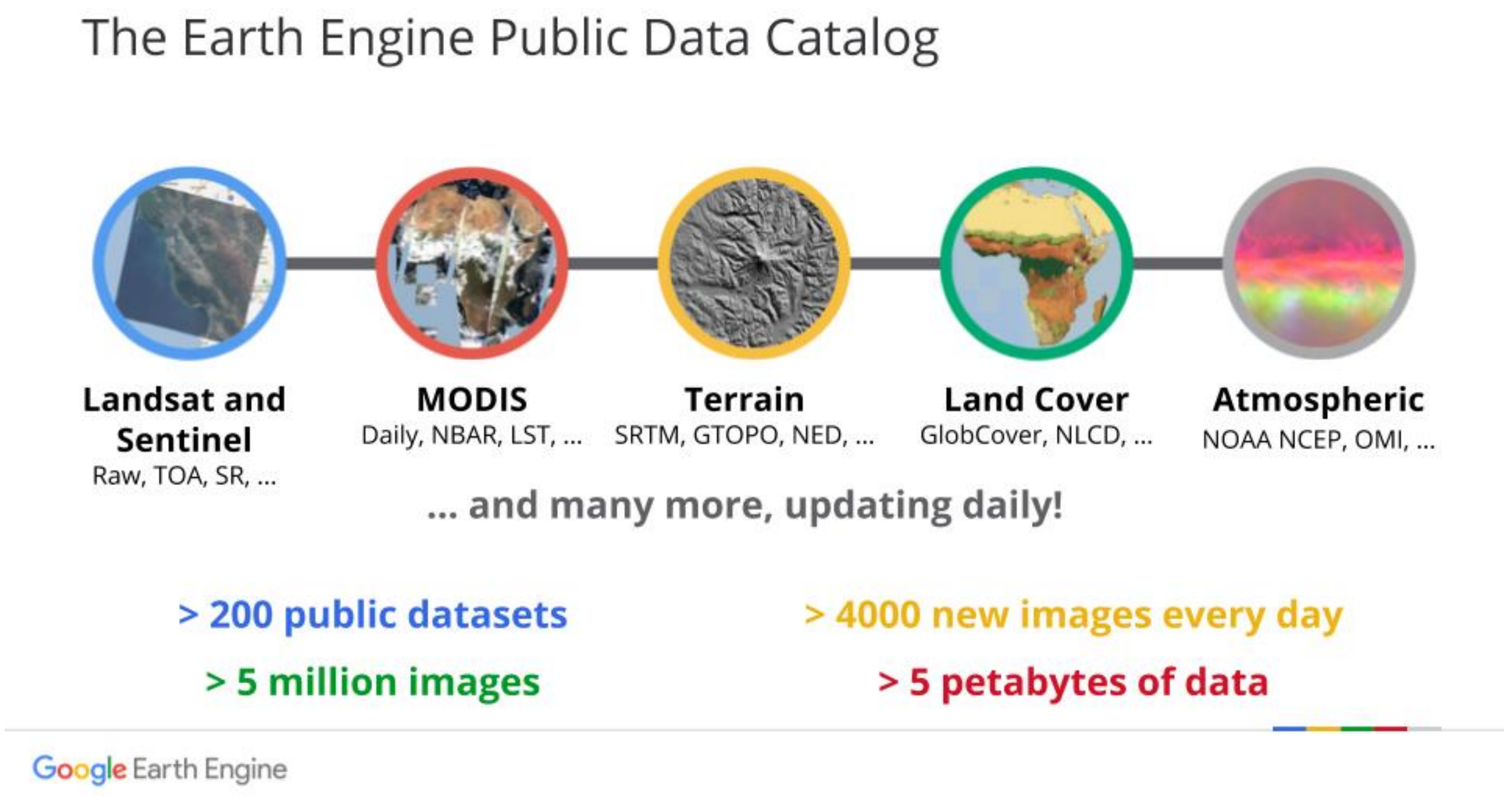
What is Google Earth Engine (GEE)?

- Google Cloud product for **geospatial** analysis at **scale**.
- Two main features:
  - Multi-petabyte **catalog** of satellite imagery and geospatial datasets
    - Can also upload own datasets
  - Planetary-scale distributed computation to accelerate analysis
- **Free** for research purposes (non-commercial use).
  - Register for Google Cloud account.
  - Register GEE project.



Very rich data catalog

- https://developers.google.com/earth-engine/datasets/catalog



The Earth Engine Public Data Catalog

Landsat and Sentinel
Raw, TOA, SR, ...

MODIS
Daily, NBAR, LST, ...

Terrain
SRTM, GTOPO, NED, ...

Land Cover
GlobCover, NLCD, ...

Atmospheric
NOAA NCEP, OMI, ...

... and many more, updating daily!

> 200 public datasets

> 4000 new images every day

> 5 million images

> 5 petabytes of data

Google Earth Engine

Distributed computation on Google server

- Avoid downloading very large datasets locally
- Load, process, and analyze directly on GEE servers
- Rich analytical capabilities
  - Spatial data processing
  - Interactive map for rapid visualizations and inspection
  - Create and export raster, vector, tabular data
  - Produce maps, figures, etc.
  - Create interactive web applications
  - Regression and ML built in (for classic models) or integrated (advanced models)

Client-side vs server-side operations

Client-side

- Javascript code that runs in your browser (or Python code that runs on your computer).
    - Your computer is performing these operations.
- Your code will have some client-side operations and objects at the highest level.
    - For example, strings defining file paths to load and save data are client-side objects.
- Think of these as the planner: organizing the structure and order of tasks that will be sent to the server.

Server-side

- Functions and objects written specifically for earth engine and optimized to be run on their server.
    - Most of the geospatial operations are done on EE servers with EE objects
    - e.g. filters, joins, aggregations, reprojections, reductions (like zonal statistics), and mapping over collections
- When you run a small amount of code in the code editor, you can print it to the console pane.
    - This sends some code to the EE server and fetches the return result to be displayed live.
    - Many tasks are too big to be sent to the server and displayed live. This will throw an error.
- Exporting sends the computation to the GEE server to run and then saves your output where you choose.
    - If data are large, may need to iterate over chunks of data or separate different steps.

Guides for getting set up with GEE

- The Google Developers platform has many useful guides and resources
  - https://developers.google.com/earth-engine/guides/getstarted
  - https://developers.google.com/earth-engine/reference/Quickstart
  - https://earthengine.google.com/
  - https://developers.google.com/earth-engine/guides/quickstart_python

- Other potentially useful resources:
  - https://courses.spatialthoughts.com/end-to-end-gee.html
  - https://github.com/opengeos/Awesome-GEE

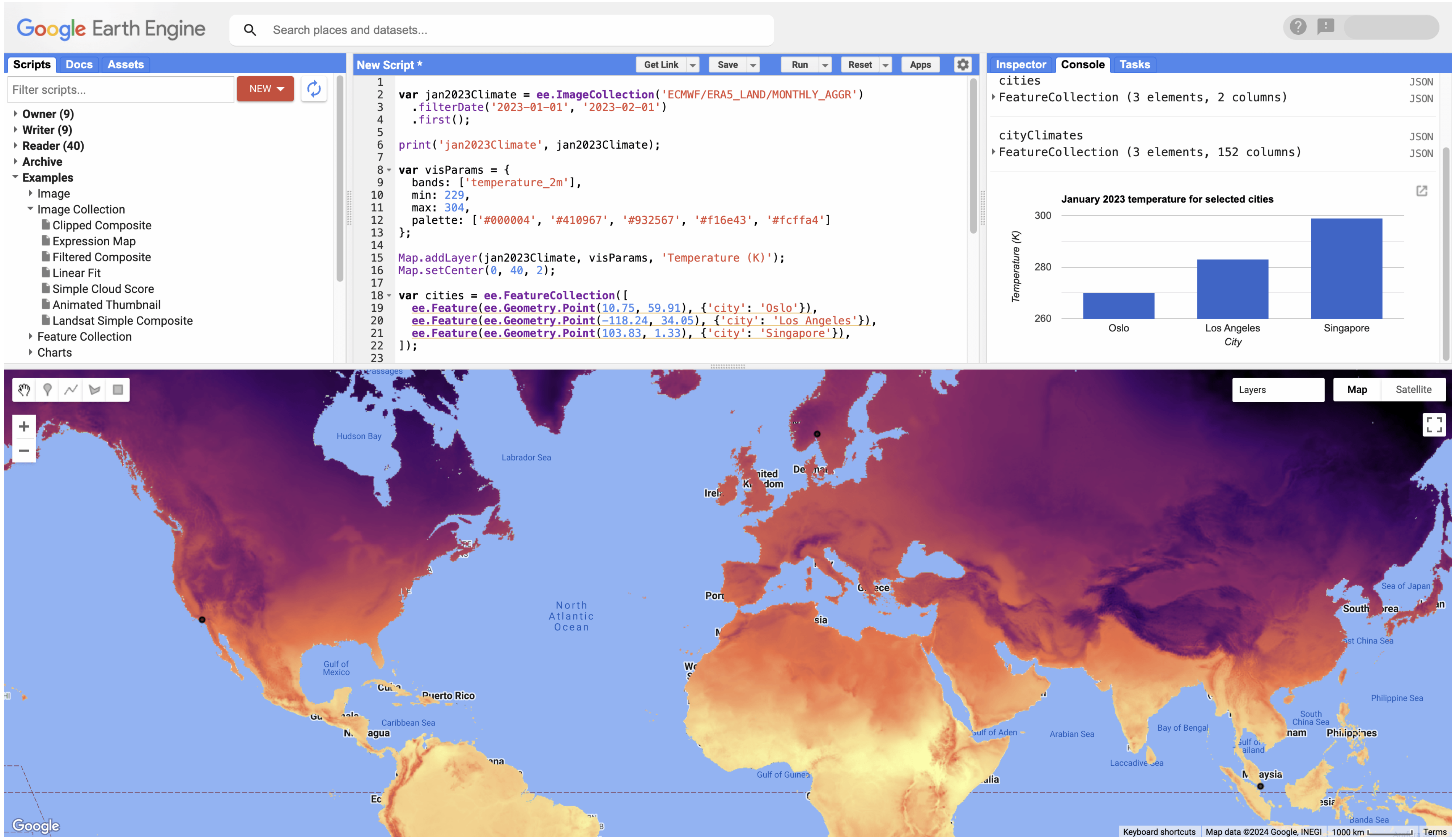
Two development environments

- **JavaScript Code Editor**: A dedicated web-based development environment for rapid prototyping, exploration, and Earth Engine App creation.
- **Python client library**: A flexible interface to Earth Engine for integration with the broader Python ecosystem, facilitating advanced workflows, and interactive analysis in Jupyter notebooks.

Why start with Java web editor?

- Access to the data catalog in the search bar, directly import data, see example scripts
- Access to the Map and Inspector: easy visualization and data checking
- Access to the Examples script folder: many example javascript files for doing most operations in EE
- Can translate Javascript code to python fairly easily
    - Much of the coding in EE uses special EE objects and functions, so the syntax for using them is pretty much identical across languages.
    - LLMs useful for generating Javascript code and Python translation

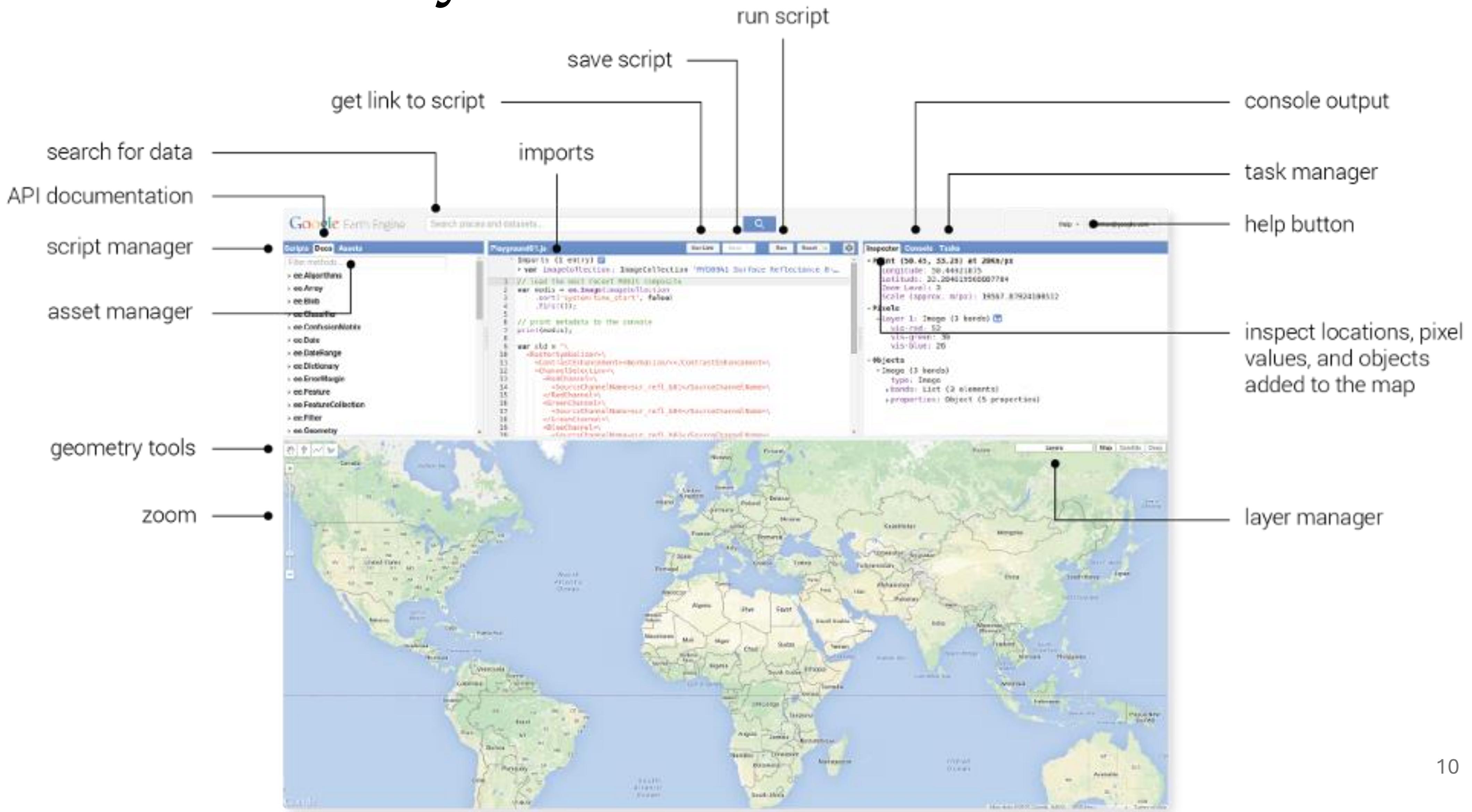
The web editor



```
var jan2023Climate = ee.ImageCollection('ECMWF/ERA5_LAND/MONTHLY_AGGR')
  .filterDate('2023-01-01', '2023-02-01')
  .first();

print('jan2023Climate', jan2023Climate);

var visParams = {
  bands: ['temperature_2m'],
  min: 229,
  max: 304,
  palette: ['#000004', '#410967', '#932567', '#f16e43', '#fcffa4']
};

Map.addLayer(jan2023Climate, visParams, 'Temperature (K)');
Map.setCenter(0, 40, 2);

var cities = ee.FeatureCollection([
  ee.Feature(ee.Geometry.Point(10.75, 59.91), {'city': 'Oslo'}),
  ee.Feature(ee.Geometry.Point(-118.24, 34.05), {'city': 'Los Angeles'}),
  ee.Feature(ee.Geometry.Point(103.83, 1.33), {'city': 'Singapore'}),
]);
```

```
cities                                              JSON
▸FeatureCollection (3 elements, 2 columns)          JSON

cityClimates                                        JSON
▸FeatureCollection (3 elements, 152 columns)        JSON
```

Web editor layout



Basic workflow

1. Identify data you want to use (e.g., 10-meter elevation data and county polygons).
2. Identify a geospatial task you need to complete (e.g., calculate average elevation and StdDev of elevation within a polygon).
3. Load the data and determine if you need to combine different datasets, and how to approach this based on their nature.
4. Filter data by date and geographical extent (e.g., the counties you will be working on).
5. Select the raster bands or table columns you will be using.
6. Build a function to do operations on a single image/raster.
    1. Any aggregation or interpolation needed (e.g., you may need to aggregate 10m elevation data to lower resolution before calculating elevation statistics on your polygons).
    2. Any temporal or spatial joins needed (e.g., combining daily and monthly data, or state and point data),
    3. Run the geospatial task on the aggregated/joined image.
7. Test the function.
    1. Run the function on a single raster or tile to check the output in the browser Map.
    2. Select a small subset of the data to test the workflow on and see if your results are what you expect.
8. Export the full task to run on the EE server.
    1. This can only export the final object to be saved somewhere.
    2. If you have many exports to make, use batch tools (e.g., if you want to save many output images)

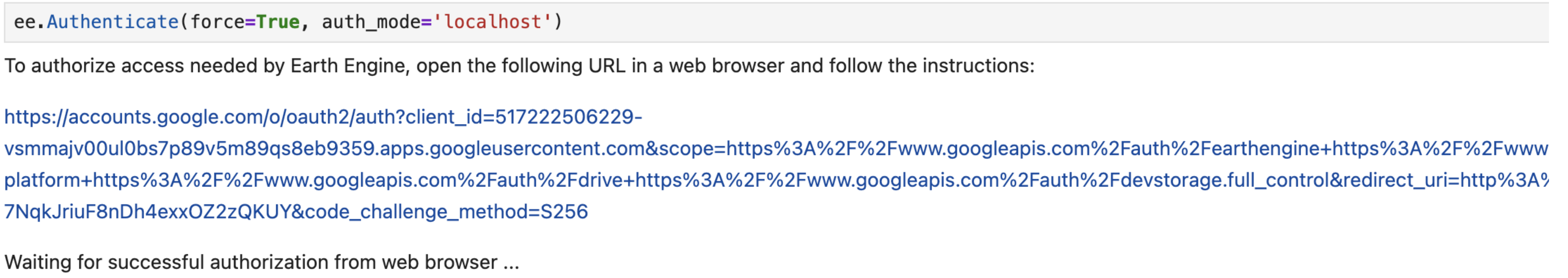
Example code in web editor

- Example code scripts
- Mapping major rivers in Chad
- Mapping fluvial flood hazard in Nigeria
- Creating a raster of monthly mean NDVI across a given area

Then, example of Python integration.

Python integration authentication

The first time you use Earth Engine on your computer, you need to authenticate it.



```
ee.Authenticate(force=True, auth_mode='localhost')
```

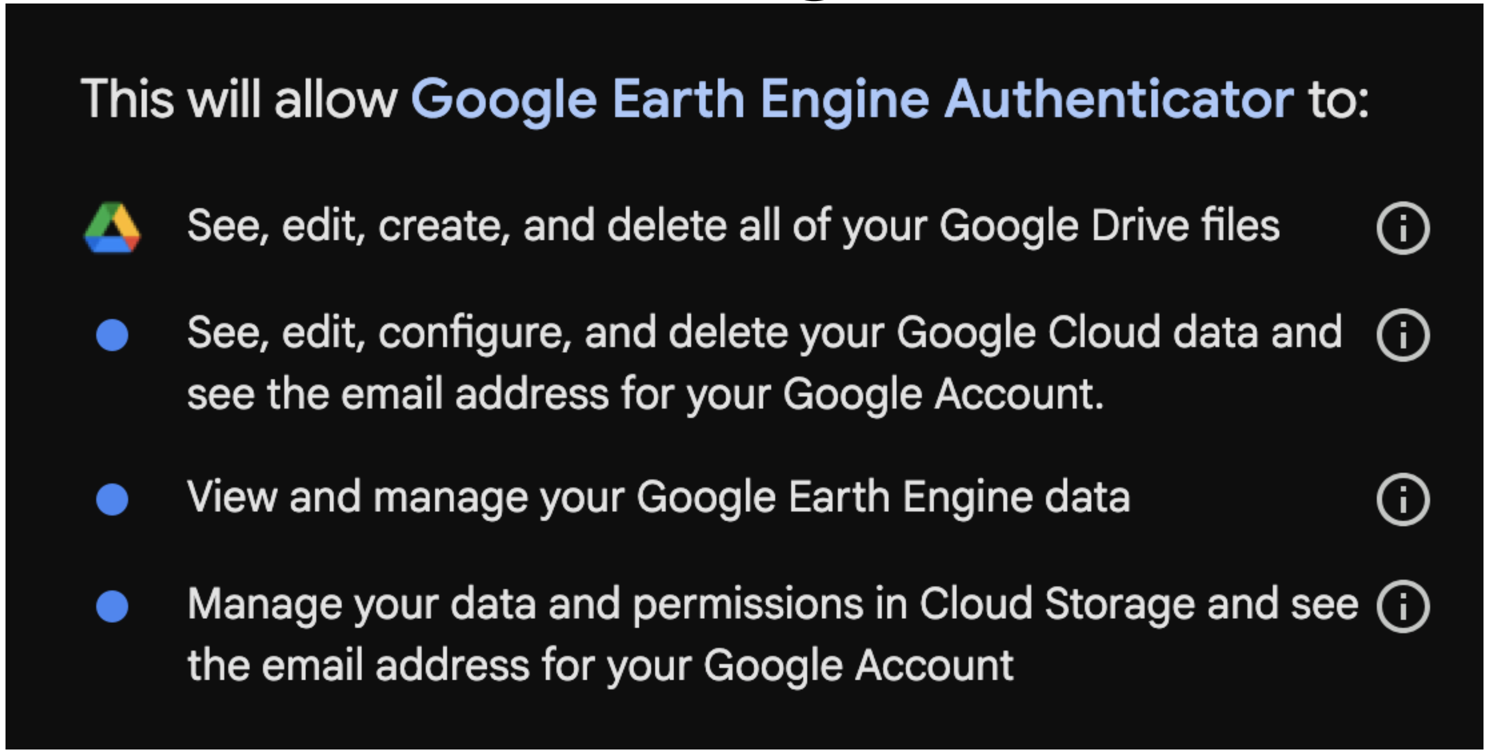
```
To authorize access needed by Earth Engine, open the following URL in a web browser and follow the instructions:

https://accounts.google.com/o/oauth2/auth?client_id=517222506229-
vsmmajv00ul0bs7p89v5m89qs8eb9359.apps.googleusercontent.com&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fearthengine+https%3A%2F%2Fwww
platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdevstorage.full_control&redirect_uri=http%3A%
7NqkJriuF8nDh4exxOZ2zQKUY&code_challenge_method=S256

Waiting for successful authorization from web browser ...
```

Installation guide: https://developers.google.com/earth-engine/guides/python_install

Reasons for using Python integration

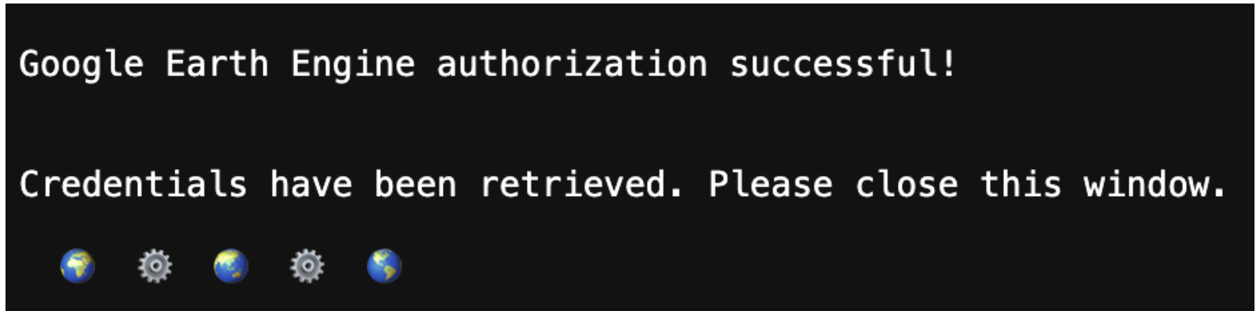
- One major issue with the javascript editor: you need to “run” export tasks for “big enough” computations”
  - Forces you to click a Run button for each task
  - Limits the number of tasks you can start
- Python allows you to programmatically start those tasks with .start()
- Useful if you:
  - Need to export many results (>100) and want all the tasks to run programmatically
  - Need to upload a lot of your own data

Allow specific permissions to your Google account, then get credentials



This will allow Google Earth Engine Authenticator to:

- See, edit, create, and delete all of your Google Drive files
- See, edit, configure, and delete your Google Cloud data and see the email address for your Google Account.
- View and manage your Google Earth Engine data
- Manage your data and permissions in Cloud Storage and see the email address for your Google Account



```
Google Earth Engine authorization successful!

Credentials have been retrieved. Please close this window.

  🌍 ⚙️ 🌏 ⚙️ 🌎
```