Global Search Trends Dashboard

Project: Visualize worldwide search interest for technology keywords using Google Trends (PyTrends), Python, and Plotly.
Visuals: World map (choropleth), treemap, heatmap, bar, line, pie & donut.
Author: *Your Name* — Aspiring Data Analyst

```
# run once
!pip install pytrends pandas plotly pycountry kaleido
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: pytrends in c:\users\asusl\appdata\
roaming\python\python312\site-packages (4.9.2)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\
site-packages (2.2.2)
Requirement already satisfied: plotly in c:\programdata\anaconda3\lib\
site-packages (5.24.1)
Requirement already satisfied: pycountry in c:\users\asusl\appdata\
roaming\python\python312\site-packages (24.6.1)
Requirement already satisfied: kaleido in c:\users\asusl\appdata\
roaming\python\python312\site-packages (1.1.0)
Requirement already satisfied: requests>=2.0 in c:\programdata\
anaconda3\lib\site-packages (from pytrends) (2.32.3)
Requirement already satisfied: lxml in c:\programdata\anaconda3\lib\
site-packages (from pytrends) (5.2.1)
Requirement already satisfied: numpy>=1.26.0 in c:\programdata\
anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\
programdata\anaconda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\programdata\
anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: tenacity>=6.2.0 in c:\programdata\
anaconda3\lib\site-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in c:\programdata\anaconda3\
lib\site-packages (from plotly) (24.1)
Requirement already satisfied: choreographer>=1.0.10 in c:\users\
asusl\appdata\roaming\python\python312\site-packages (from kaleido)
(1.2.0)
Requirement already satisfied: logistro>=1.0.8 in c:\users\asusl\
appdata\roaming\python\python312\site-packages (from kaleido) (2.0.0)
Requirement already satisfied: orjson>=3.10.15 in c:\users\asusl\
appdata\roaming\python\python312\site-packages (from kaleido) (3.11.4)
Requirement already satisfied: pytest-timeout>=2.4.0 in c:\users\
asusl\appdata\roaming\python\python312\site-packages (from kaleido)
```

```
(2.4.0)
Requirement already satisfied: simplejson>=3.19.3 in c:\users\asusl\
appdata\roaming\python\python312\site-packages (from
choreographer>=1.0.10->kaleido) (3.20.2)
Requirement already satisfied: pytest>=7.0.0 in c:\programdata\
anaconda3\lib\site-packages (from pytest-timeout>=2.4.0->kaleido)
(7.4.4)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\
lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\
programdata\anaconda3\lib\site-packages (from requests>=2.0->pytrends)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\
anaconda3\lib\site-packages (from requests>=2.0->pytrends) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\
anaconda3\lib\site-packages (from requests>=2.0->pytrends) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\
anaconda3\lib\site-packages (from requests>=2.0->pytrends) (2024.8.30)
Requirement already satisfied: iniconfig in c:\programdata\anaconda3\
lib\site-packages (from pytest>=7.0.0->pytest-timeout>=2.4.0->kaleido)
(1.1.1)
Requirement already satisfied: pluggy<2.0,>=0.12 in c:\programdata\
anaconda3\lib\site-packages (from pytest>=7.0.0->pytest-
timeout>=2.4.0->kaleido) (1.0.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\
lib\site-packages (from pytest>=7.0.0->pytest-timeout>=2.4.0->kaleido)
(0.4.6)
```

```python
# Cell 1: imports and small helpers
from pytrends.request import TrendReq
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
import pycountry
import time, os, json

# helper to map country name -> ISO3
def country_to_iso3(name, manual_fixes=None):
    if manual_fixes is None:
        manual_fixes = {}
    lookup = manual_fixes.get(name, name)
    try:
        return pycountry.countries.lookup(lookup).alpha_3
    except Exception:
        try:
            return pycountry.countries.search_fuzzy(lookup)[0].alpha_3
        except Exception:
            return None
```

```
# small pretty-print
def s(text):
    print("\n" + "="*6 + " " + str(text) + " " + "="*6 + "\n")

# Cell 2: configuration
keywords = ["python", "pandas", "power bi", "machine learning"]   #
change to your keywords
timeframe = 'today 12-m'   # 'today 12-m' (last 12 months) - change if
needed
geo = ''                   # '' = worldwide, or 'IN' for India, etc.

# cache filenames (used to avoid re-querying Google)
CACHE_IOT = 'pytrends_interest_over_time.csv'
CACHE_BY_REGION = 'pytrends_interest_by_region.csv'
CACHE_PER_KEYWORD = 'pytrends_iot_per_keyword.csv'

# Cell 3: fetch by_region and interest_over_time with retries and
caching
s("Initializing pytrends session")
pytrends = TrendReq(hl='en-US', tz=330)   # tz=330 -> Asia/Kolkata

# 1) Get interest_by_region (country-level)
s("Fetching interest_by_region (country-level)")
try:
    pytrends.build_payload(keywords, timeframe=timeframe, geo=geo)
    by_region = pytrends.interest_by_region(resolution='COUNTRY',
inc_low_vol=True, inc_geo_code=False)
    # normalize: ensure a 'country' column even if returned as index
    if 'geoName' in by_region.columns:
        by_region =
by_region.reset_index().rename(columns={'geoName':'country'})
    else:
        by_region =
by_region.reset_index().rename(columns={by_region.index.name or
0:'country'})
    # map ISO3
    manual_fixes = {"UK":"United Kingdom","Viet
Nam":"Vietnam","Russia":"Russian Federation","Czechia":"Czech
Republic"}
    by_region['iso_alpha'] = by_region['country'].apply(lambda x:
country_to_iso3(x, manual_fixes))
    by_region.to_csv(CACHE_BY_REGION, index=False)
    print("by_region fetched and cached.")
except Exception as e:
    print("Failed to fetch by_region via pytrends:", e)
    if os.path.exists(CACHE_BY_REGION):
        print("Loading cached by_region from file.")
        by_region = pd.read_csv(CACHE_BY_REGION)
    else:
        print("No cached by_region found. Creating a small demo
```

```
by_region.")
        demo_countries = ['India','United States','United
Kingdom','Canada','Germany','Brazil','Australia','France','Japan','Ind
onesia']
        by_region = pd.DataFrame({'country': demo_countries})
        np.random.seed(0)
        for k in keywords:
            by_region[k] =
np.random.randint(20,100,size=len(demo_countries))
        by_region['iso_alpha'] = by_region['country'].apply(lambda x:
country_to_iso3(x) or None)

# 2) Try to get interest_over_time (iot). If 429 -> fallback to cache;
if none -> synth.
def fetch_iot_with_retries(keywords, timeframe, geo, attempts=4):
    attempt = 0
    wait = 1
    while attempt < attempts:
        try:
            pytrends.build_payload(keywords, timeframe=timeframe,
geo=geo)
            iot = pytrends.interest_over_time()
            if iot is None or iot.empty:
                raise ValueError("empty interest_over_time result")
            if 'isPartial' in iot.columns:
                iot = iot.drop(columns=['isPartial'])
            iot = iot.reset_index().rename(columns={'date':'Date'})
            iot.to_csv(CACHE_IOT, index=False)
            return iot
        except Exception as e:
            print(f"Attempt {attempt+1} failed: {e}. Backing off
{wait}s")
            time.sleep(wait)
            wait *= 2
            attempt += 1
    # exhausted
    if os.path.exists(CACHE_IOT):
        print("Loading cached interest_over_time CSV.")
        try:
            return pd.read_csv(CACHE_IOT, parse_dates=['Date'])
        except:
            return None
    return None

s("Fetching interest_over_time (time-series)")
iot = fetch_iot_with_retries(keywords, timeframe, geo, attempts=4)

# If iot is None, try per-keyword slow fetch (optional) OR build
synthetic monthly from by_region means
if iot is None:
```

```
    s("interest_over_time not available. Creating synthetic monthly
data based on by_region averages.")
    available = [k for k in keywords if k in by_region.columns]
    if not available:
        available = keywords.copy()
    base_levels = {k: float(by_region[k].mean()) if k in
by_region.columns else 50.0 for k in available}
    months = pd.date_range(end=pd.Timestamp.today(), periods=12,
freq='M').to_period('M').to_timestamp()
    synth = pd.DataFrame({'Month': months})
    np.random.seed(42)
    for k in available:
        season = (np.sin(np.linspace(0, 2*np.pi, len(months)) - 0.5) +
1) / 2
        trend = np.linspace(0.95,1.05,len(months))
        noise = np.random.normal(0,0.08,size=len(months))
        series = season*0.6 + trend*0.4 + noise
        series = (series - series.min())/(series.max()-series.min()
+1e-9)
        scaled = np.clip(series * base_levels.get(k,50), 0,
100).round(2)
        synth[k] = scaled
    monthly = synth.copy()
    keywords = available
else:
    # build monthly from real iot
    iot['Month'] =
pd.to_datetime(iot['Date']).dt.to_period('M').dt.to_timestamp()
    numeric_cols = [c for c in keywords if c in iot.columns]
    if not numeric_cols:
        numeric_cols =
iot.select_dtypes(include='number').columns.tolist()
    monthly = iot.groupby('Month')[numeric_cols].mean().reset_index()
    keywords = numeric_cols

s("Data ready (real or synthetic).")
display(by_region.head())
display(monthly.head())


====== Initializing pytrends session ======


====== Fetching interest_by_region (country-level) ======

Failed to fetch by_region via pytrends: The request failed: Google
returned a response with code 429
Loading cached by_region from file.

====== Fetching interest_over_time (time-series) ======
```

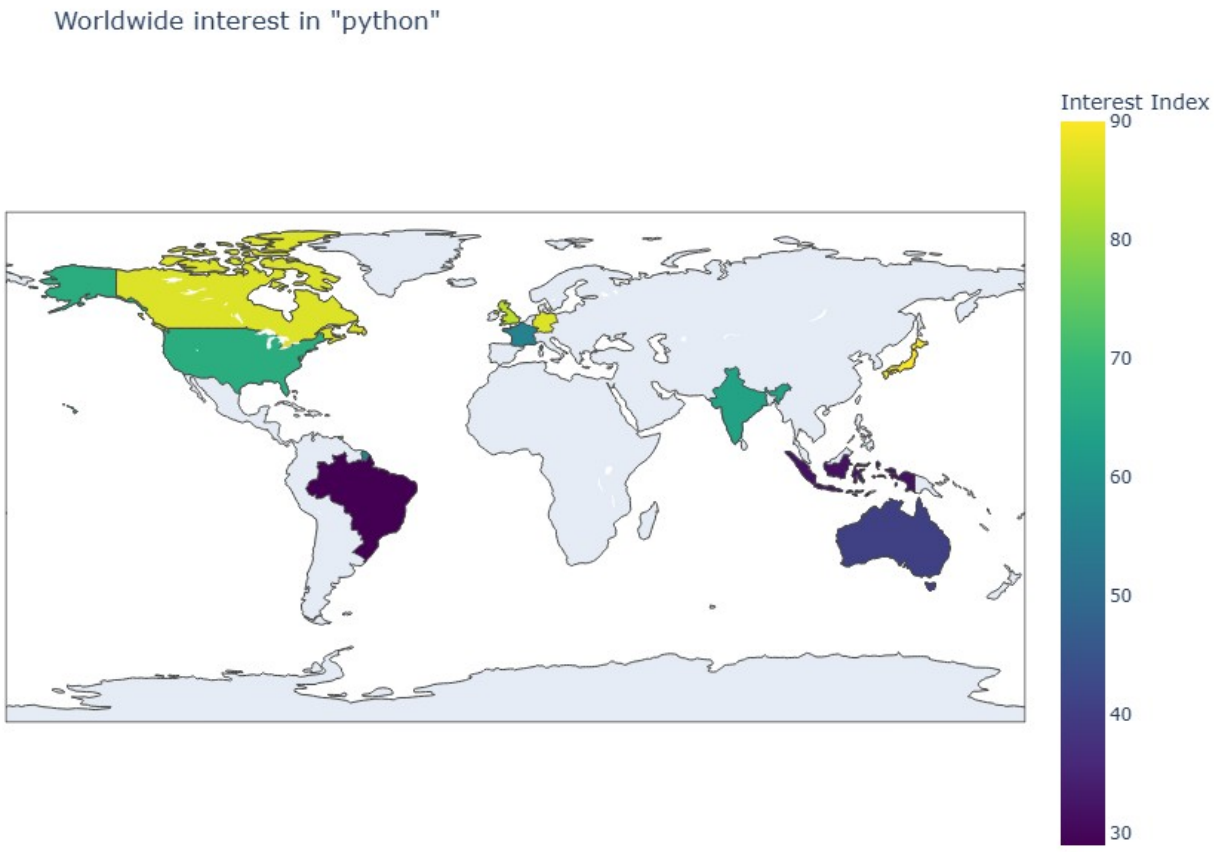
```
Attempt 1 failed: The request failed: Google returned a response with
code 429. Backing off 1s
Attempt 2 failed: The request failed: Google returned a response with
code 429. Backing off 2s
Attempt 3 failed: The request failed: Google returned a response with
code 429. Backing off 4s
Attempt 4 failed: The request failed: Google returned a response with
code 429. Backing off 8s

====== interest_over_time not available. Creating synthetic monthly
data based on by_region averages. ======


====== Data ready (real or synthetic). ======


C:\Users\asusl\AppData\Local\Temp\ipykernel_23720\1816708782.py:73:
FutureWarning: 'M' is deprecated and will be removed in a future
version, please use 'ME' instead.
  months = pd.date_range(end=pd.Timestamp.today(), periods=12,
freq='M').to_period('M').to_timestamp()

          country  python  pandas  power bi  machine learning
iso_alpha
0           India      64      78        89                52
IND
1   United States      67      85        99                85
USA
2  United Kingdom      84      59        67                29
GBR
3          Canada      87      66        84                77
CAN
4         Germany      87      57        69                52
DEU

       Month  python  pandas  power bi  machine learning
0 2024-11-01   15.03   19.21     15.28             18.39
1 2024-12-01   26.12   18.78     35.92             17.96
2 2025-01-01   47.26   40.03     41.61             37.29
3 2025-02-01   63.70   63.78     62.80             57.93
4 2025-03-01   51.97   61.34     56.98             63.30
```

```python
# Cell 4: top 10 text output (no chart)
top_keyword = 'python' if 'python' in keywords else keywords[0]
s(f"Top 10 countries by interest for: {top_keyword}")
top_10 = by_region.sort_values(by=top_keyword,
ascending=False).head(10)
display(top_10[['country', top_keyword]].reset_index(drop=True))
```

```
====== Top 10 countries by interest for: python ======


          country  python
0           Japan      90
1          Canada      87
2         Germany      87
3  United Kingdom      84
4   United States      67
5           India      64
6          France      56
7       Australia      41
8       Indonesia      32
9          Brazil      29
```

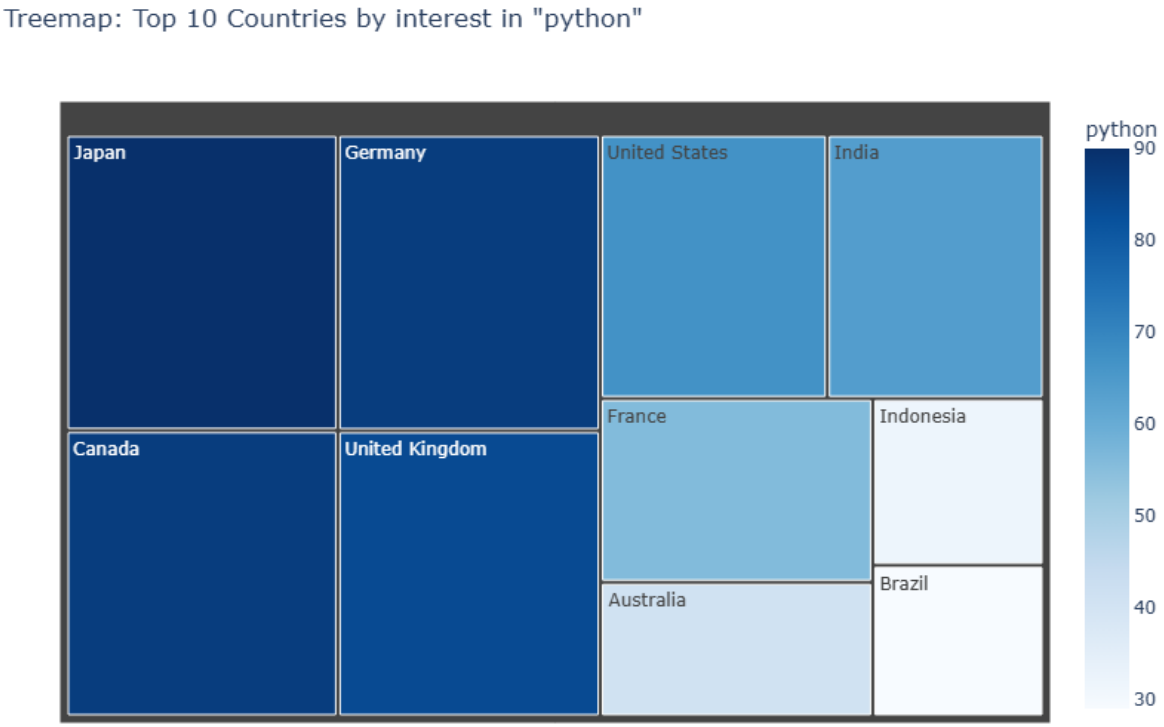
```
# Cell 5: Choropleth map
s("Choropleth (world map)")
chor_df = by_region.copy()
use_iso = ('iso_alpha' in chor_df.columns) and
(chor_df['iso_alpha'].notna().sum() >= max(5, 0.5*len(chor_df)))

if use_iso:
    fig_map = px.choropleth(
        chor_df,
        locations='iso_alpha',
        color=top_keyword,
        hover_name='country',
        title=f'Worldwide interest in "{top_keyword}"',
        color_continuous_scale='Viridis',
        labels={top_keyword:'Interest Index'},
        locationmode='ISO-3'
    )
else:
    fig_map = px.choropleth(
        chor_df,
        locations='country',
        color=top_keyword,
        hover_name='country',
        title=f'Worldwide interest in "{top_keyword}" (country
names)',
        color_continuous_scale='Viridis',
        labels={top_keyword:'Interest Index'},
        locationmode='country names'
    )

fig_map.update_layout(height=600, margin=dict(t=60,l=10,r=10,b=10))
fig_map.show()
```

```
====== Choropleth (world map) ======


C:\Users\asusl\AppData\Roaming\Python\Python312\site-packages\kaleido\
_sync_server.py:11: UserWarning:


Warning: You have Plotly version 5.24.1, which is not compatible with
this version of Kaleido (1.1.0).

This means that static image generation (e.g. `fig.write_image()`)
will not work.

Please upgrade Plotly to version 6.1.1 or greater, or downgrade
Kaleido to version 0.2.1.
```

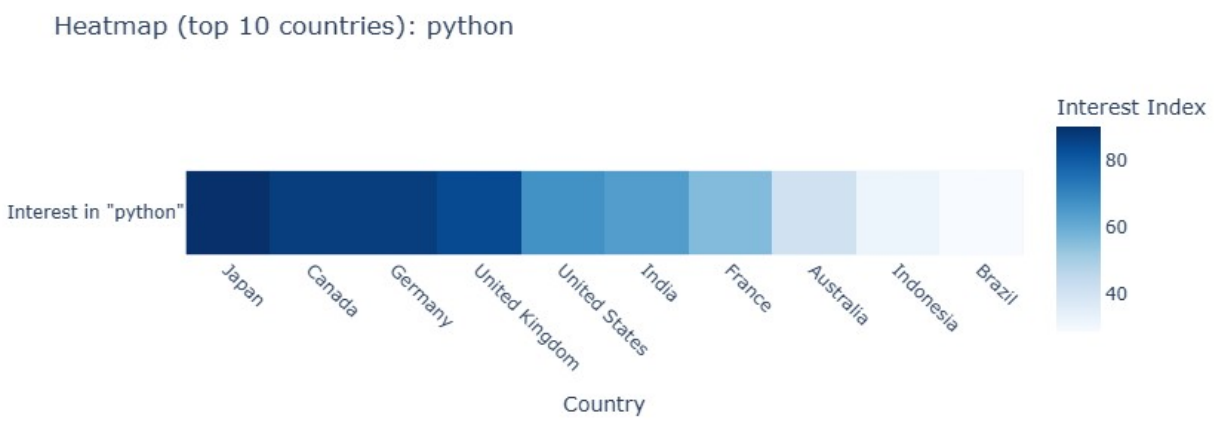



```
# Cell 6: Treemap (top N)
s("Treemap")
top_n = 10
```

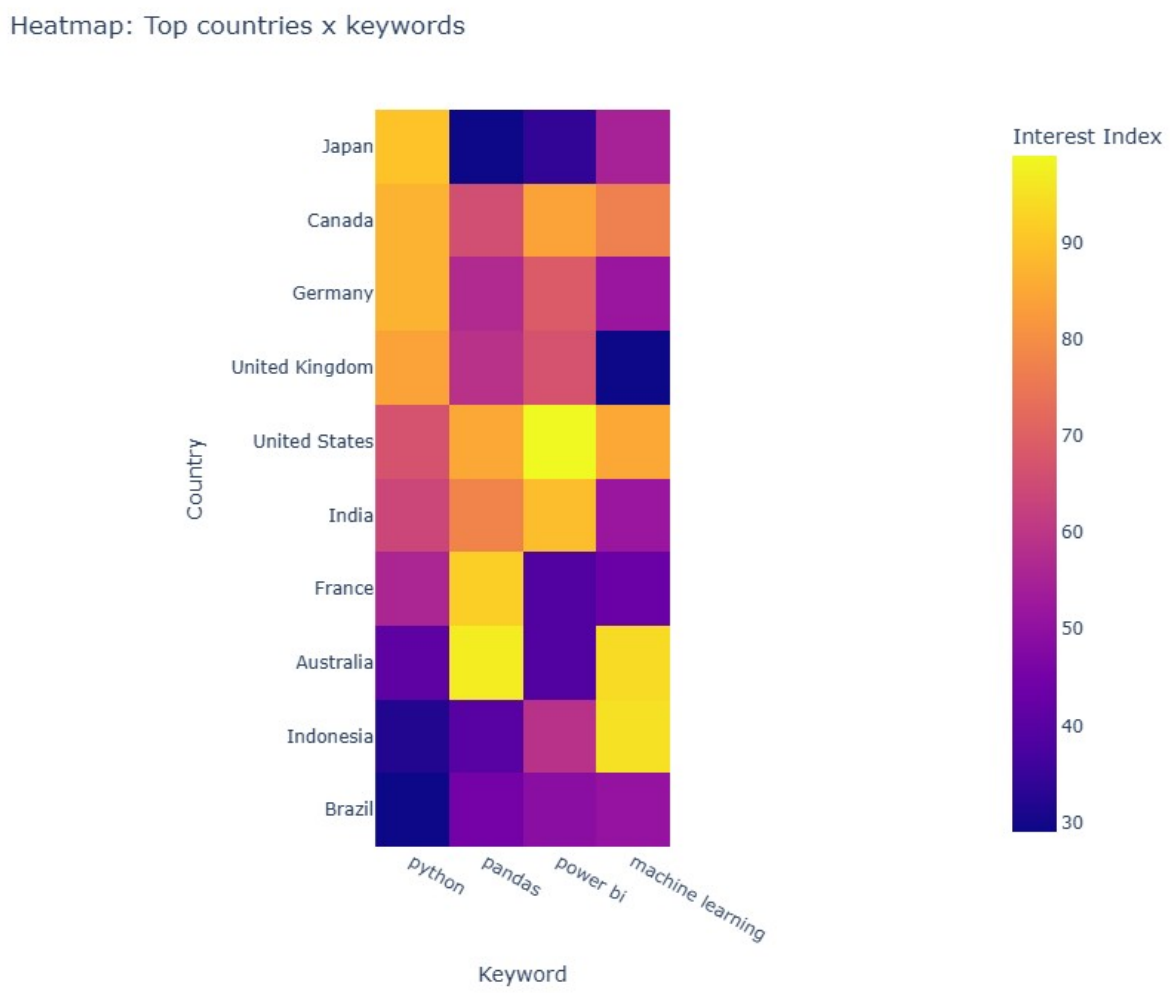
```
treemap_df = by_region.sort_values(by=top_keyword,
ascending=False).head(top_n).copy()
# if all equal, add tiny jitter
if treemap_df[top_keyword].nunique() == 1:
    treemap_df[top_keyword] = treemap_df[top_keyword] +
np.linspace(0,0.001,len(treemap_df))
fig_treemap = px.treemap(
    treemap_df,
    path=['country'],
    values=top_keyword,
    color=top_keyword,
    color_continuous_scale='Blues',
    title=f'Treemap: Top {top_n} Countries by interest in
"{top_keyword}"'
)
fig_treemap.update_layout(height=600)
fig_treemap.show()


====== Treemap ======
```



```
# Cell 7: Heatmap variants
s("Heatmap: 1-row by country")
top_n = 10
heat_df = by_region.sort_values(by=top_keyword,
ascending=False).head(top_n)
fig_heat = px.imshow(
    [heat_df[top_keyword].values],
    labels=dict(x="Country", y="", color="Interest Index"),
    x=heat_df['country'],
    y=[f'Interest in "{top_keyword}"'],
    color_continuous_scale='Blues'
)
fig_heat.update_layout(title=f'Heatmap (top {top_n} countries):
{top_keyword}', xaxis_tickangle=45, height=320)
fig_heat.show()

s("Heatmap: matrix countries x keywords")
available_keys = [k for k in keywords if k in by_region.columns]
if available_keys:
    top_countries = by_region.sort_values(by=available_keys[0],
ascending=False).head(20)['country']
    matrix_df = by_region.set_index('country').loc[top_countries,
available_keys].fillna(0)
    fig_matrix = px.imshow(matrix_df, labels=dict(x="Keyword",
y="Country", color="Interest Index"),
                           title=f'Heatmap: Top countries x keywords')
    fig_matrix.update_layout(height=700)
    fig_matrix.show()
else:
    print("No keyword columns available for matrix heatmap.")


====== Heatmap: 1-row by country ======
```



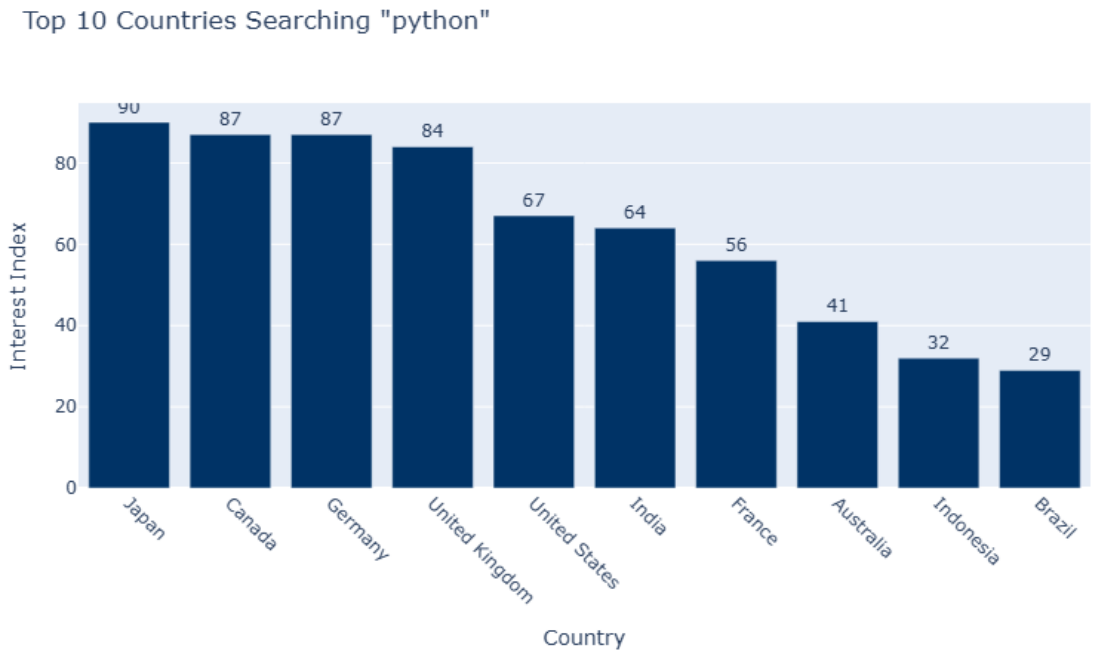
```
====== Heatmap: matrix countries x keywords ======
```




```
# Cell 8: Vertical bar chart
s("Vertical bar chart")
top_n = 10
bar_df = by_region.sort_values(by=top_keyword,
ascending=False).head(top_n).copy()
fig_bar = px.bar(
    bar_df,
    x='country',
    y=top_keyword,
    title=f'Top {top_n} Countries Searching "{top_keyword}"',
    labels={top_keyword:'Interest Index', 'country':'Country'},
    color_discrete_sequence=['#003366']  # dark blue
)
fig_bar.update_layout(xaxis_tickangle=45, height=520,
```

```
margin=dict(b=160))
fig_bar.update_traces(text=bar_df[top_keyword],
textposition='outside')
fig_bar.show()
```

```
====== Vertical bar chart ======
```




```
# Cell 9: Line chart (monthly trends)
s("Line chart: monthly trends")
if 'Month' not in monthly.columns:
    monthly['Month'] = pd.date_range(end=pd.Timestamp.today(),
periods=len(monthly), freq='M')
fig_line = px.line(monthly, x='Month', y=keywords,
                   title='Monthly Search Interest (index)',
                   labels={'value':'Interest Index', 'Month':'Month'})
fig_line.update_layout(legend_title_text='Keyword', height=520)
fig_line.show()
```

```
====== Line chart: monthly trends ======
```


```
# Cell 10: Pie & Donut charts
s("Pie & Donut")
pie_df = by_region.sort_values(by=top_keyword,
ascending=False).head(8).copy()
if pie_df[top_keyword].sum() == 0:
    pie_df[top_keyword] = np.arange(len(pie_df),0,-1)
pie_df['pct'] = (pie_df[top_keyword] / pie_df[top_keyword].sum() *
100).round(2)

fig_pie = px.pie(pie_df, names='country', values='pct',
                 title=f'Pie: Top 8 Countries for "{top_keyword}"',
hole=0.0)
fig_pie.update_traces(textposition='inside', textinfo='percent+label')
fig_pie.show()

fig_donut = px.pie(pie_df, names='country', values='pct',
                   title=f'Donut: Top 8 Countries for
"{top_keyword}"', hole=0.45)
fig_donut.update_traces(textposition='inside',
textinfo='percent+label')
fig_donut.show()


====== Pie & Donut ======
```

Pie: Top 8 Countries for "python"

Donut: Top 8 Countries for "python"


```
# Cell 11: Save outputs (CSV + static images)
s("Saving CSV outputs")
by_region.to_csv(CACHE_BY_REGION, index=False)
monthly.to_csv('pytrends_monthly.csv', index=False)
print("Saved CSVs.")

# Save images (requires kaleido)
try:
    fig_map.write_image("choropleth.png", scale=2)
    fig_treemap.write_image("treemap.png", scale=2)
    fig_bar.write_image("bar_top_countries.png", scale=2)
    fig_line.write_image("line_monthly.png", scale=2)
    fig_pie.write_image("pie_top8.png", scale=2)
    print("Saved figure PNGs.")
except Exception as e:
    print("Could not write PNGs (kaleido missing or fig not
defined).", e)
```

```
====== Saving CSV outputs ======

Saved CSVs.
Could not write PNGs (kaleido missing or fig not defined).
Image export using the "kaleido" engine requires the kaleido package,
which can be installed using pip:
    $ pip install -U kaleido
```