METADATA

- **Data source:** ERA5 hourly data on single levels from 1940 to present
- **Product type:** Monthly averaged reanalysis by hour of day
- **Variable:** Boundary layer height
- **Year:** 2024
- **Month:** June
- **Days:** 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
- **Time:** 00:00, 01:00, 02:00, 03:00, 04:00, 05:00, 06:00, 07:00, 08:00, 09:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00, 17:00, 18:00, 19:00, 20:00, 21:00, 22:00, 23:00
- **Whole available region:** Across Ghana [North 11.5°, West -3.5°, South 0°, East 1.5°]
- **Format:** NetCDF (experimental)

In [ ]:

In [ ]:

IMPORTING ALL THE REQUIRED LIBRARIES

In [2]:
```
import xarray as xr
import matplotlib.pyplot as plt
import numpy as np
import cartopy.crs as ccrs
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import cartopy.feature as cfeature
```

In [15]:
```
data = xr.open_dataset('blh data.nc')

Kumasi_da = data.sel(longitude=slice(-1.5,3.5),latitude=slice(11.5,4.5))
Kumasi_da
```

Out[15]: xarray.Dataset

► Dimensions: (**longitude**: 13, **latitude**: 29, **time**: 720)

▼ Coordinates:

longitude	(longitude)	float32	-1.5 -1.25 ...
latitude	(latitude)	float32	11.5 11.25...
time	(time)	datetime64[ns]	2024-06-0...

▼ Data variables:

blh	(time, latitude, longitude)	float32	...

► Indexes: (3)

▼ Attributes:

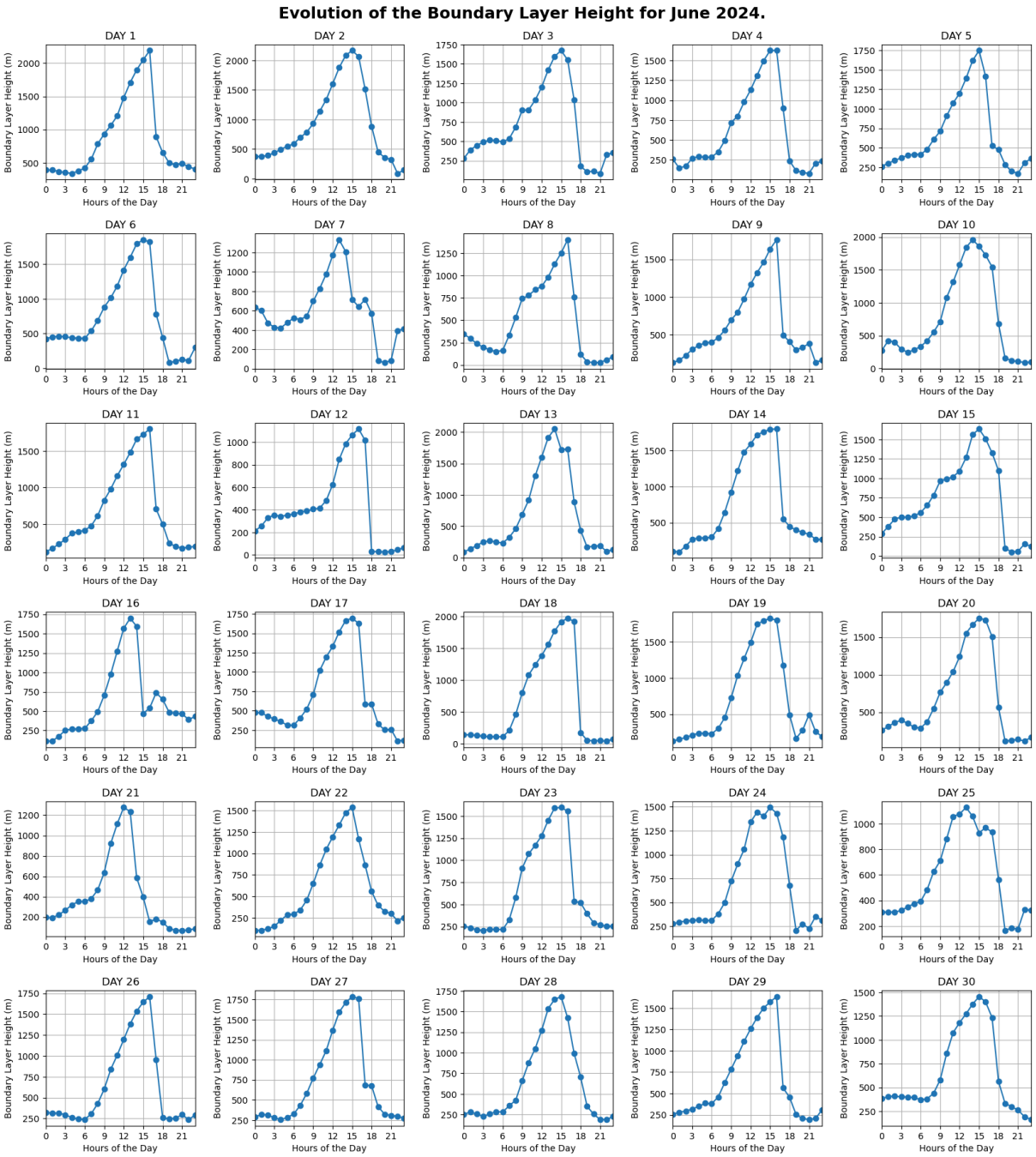
Conventions :	CF-1.6
history :	2024-07-08 11:59:46 GMT by grib_to_netcdf-2.28.1: /opt/ecmwf/mars-client/bin/grib_to_netcdf -S param -o /cache/data9/adaptor.mars.internal-1720439984.8376882-16869-1-89c1e514-3beb-43cf-a5d1-731cb30f1d0c.nc /cache/tmp/89c1e514-3beb-43cf-a5d1-731cb30f1d0c-adaptor.mars.internal-1720439927.24917-16869-1-tmp.grib

In [24]:
```
# Extract BLH data for a single location
boundary_layer_height = Kumasi_da['blh'][:, 0, 0]     #data['blh']:it Selects

#[:, 0, 0]: Selects all time steps (:) for the first latitude (0) and the firs
#This assumes the dataset has dimensions in the order of time, latitude, and l



# Convert time to a more readable format
time = data['time'].values   #data['time'].values: Extracts the time variable

# Select three different days for plotting
selected_days = np.arange(0, len(time), len(time) // 30)[:30]
selected_days
```

Out[24]:
```
array([  0,  24,  48,  72,  96, 120, 144, 168, 192, 216, 240, 264, 288,
       312, 336, 360, 384, 408, 432, 456, 480, 504, 528, 552, 576, 600,
       624, 648, 672, 696])
```

In [41]:
```
# Plotting
fig, axes = plt.subplots(6,5, figsize=(20, 20))

# Flatten the axes array for easier iteration
axes = axes.flatten()
```

```
for i, day in enumerate(selected_days):
    # Extract a 24-hour period, assuming each time step is an hour
    start_idx = day
    end_idx = day + 24

    # Check if the end index exceeds the length of the data
    if end_idx > len(boundary_layer_height):
        end_idx = len(boundary_layer_height)
        start_idx = end_idx - 24 if end_idx - 24 >= 0 else 0

    hours = np.arange(0, end_idx - start_idx)
    blh_day = boundary_layer_height[start_idx:end_idx]

    ax = axes[i]
    ax.plot(hours, blh_day, marker='o', linestyle='-')
    ax.set_title(f'DAY {i+1}')
    ax.set_xlabel('Hours of the Day')
    ax.set_ylabel('Boundary Layer Height (m)')
    ax.grid(True)
    ax.set_xlim(0, 23)  # Set x-axis limits to show hours from 00:00 to 23:00
    ax.set_xticks(np.arange(0, 24, 3))  # Set x-axis ticks to show every 3 hou

# Hide any unused subplots
for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])


plt.suptitle("Evolution of the Boundary Layer Height for June 2024.",ha = 'cen
fig.subplots_adjust(left=0.125, bottom=0.1, right=0.9, top=0.95, wspace=0.4, h
# Adjust layout
# plt.tight_layout()
plt.show()
```


Evolution of the Boundary Layer Height for June 2024.
DAY 1
DAY 2
DAY 3
DAY 4
DAY 5
DAY 6
DAY 7
DAY 8
DAY 9
DAY 10
DAY 11
DAY 12
DAY 13
DAY 14
DAY 15
DAY 16
DAY 17
DAY 18
DAY 19
DAY 20
DAY 21
DAY 22
DAY 23
DAY 24
DAY 25
DAY 26
DAY 27
DAY 28
DAY 29
DAY 30
Boundary Layer Height (m)
Hours of the Day

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: