```python
import numpy as np
import matplotlib.pyplot as plt
import xarray as xr
import pandas as pd
import cartopy.crs as ccrs
import cartopy.feature as cfeature

# □□□□
file_path = "200301_202006-C3S-L3_GHG-PRODUCTS-OBS4MIPS-MERGED-v4.3.nc"
ds = xr.open_dataset(file_path)
methane = ds['xch4']

print("Dataset loaded successfully:")
print(methane)


# Task 1.1: Compute methane climatology for each month
def plot_methane_climatology():
    # Calculate monthly climatology
    monthly_climatology = methane.groupby('time.month').mean(dim='time')

    # Create figure with 12 subplots (3x4 grid)
    fig, axes = plt.subplots(3, 4, figsize=(20, 15),
                             subplot_kw={'projection': ccrs.PlateCarree()})
    axes = axes.flatten()

    month_names = ['January', 'February', 'March', 'April', 'May', 'June',
                   'July', 'August', 'September', 'October', 'November', 'December']

    # Plot each month
    for i, month in enumerate(range(1, 13)):
        ax = axes[i]
        monthly_data = monthly_climatology.sel(month=month)

```

```
        # Create contour plot
        contour = monthly_data.plot.contourf(ax=ax, levels=20,
                                             transform=ccrs.PlateCarree(),
                                             cmap='viridis', add_colorbar=False)

        # Add coastlines and gridlines
        ax.coastlines()
        ax.gridlines(draw_labels=True, linestyle='--', alpha=0.7)
        ax.set_title(f'{month_names[i]} Climatology', fontsize=12, fontweight='bold')

        # Add features
        ax.add_feature(cfeature.BORDERS, linestyle=':', alpha=0.5)

    # Add colorbar
    plt.subplots_adjust(right=0.9)
    cbar_ax = fig.add_axes([0.92, 0.15, 0.02, 0.7])
    fig.colorbar(contour, cax=cbar_ax, label='Methane (ppb)')

    plt.suptitle('Methane Climatology (2003-2020) by Month', fontsize=16, fontweight='bold')
    plt.tight_layout()
    plt.savefig('methane_climatology.png', dpi=300, bbox_inches='tight')
    plt.show()

    return monthly_climatology


# □□□□□□□
monthly_climatology = plot_methane_climatology()


# Task 1.2: Plot globally-averaged methane time series
```

```python
def plot_global_methane_timeseries():
    # Calculate global average (area-weighted)
    # First, we need to account for different grid cell areas due to latitude
    lat_weights = np.cos(np.deg2rad(methane.lat))

    # Calculate global average for each time step
    global_avg = methane.weighted(lat_weights).mean(dim=['lat', 'lon'])

    # Convert to pandas Series for easier plotting
    time_index = pd.to_datetime(global_avg.time.values)
    global_series = pd.Series(global_avg.values, index=time_index)

    # Create plot
    plt.figure(figsize=(12, 6))
    plt.plot(global_series.index, global_series.values, linewidth=2, color='red')

    # Add trend line
    z = np.polyfit(range(len(global_series)), global_series.values, 1)
    p = np.poly1d(z)
    plt.plot(global_series.index, p(range(len(global_series))),
             '--', color='black', linewidth=1.5, label='Linear Trend')

    # Format plot
    plt.title('Globally-Averaged Methane Levels (2003-2020)', fontsize=14, fontweight='bold')
    plt.xlabel('Year')
    plt.ylabel('Methane (ppb)')
    plt.grid(True, alpha=0.3)
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig('global_methane_timeseries.png', dpi=300, bbox_inches='tight')
```

```
    plt.show()

    # Print statistics
    print("Global Methane Analysis:")
    print(f"Start value (2003): {global_series.iloc[0]:.1f} ppb")
    print(f"End value (2020): {global_series.iloc[-1]:.1f} ppb")
    print(f"Total increase: {global_series.iloc[-1] - global_series.iloc[0]:.1f} ppb")
    print(f"Average annual increase: {(global_series.iloc[-1] - global_series.iloc[0]) / 17.5:.1f} ppb/year")

    return global_series


# Execute global average analysis
global_methane = plot_global_methane_timeseries()


# Task 1.3: Plot deseasonalized methane at [15°S, 150°W]
def plot_deseasonalized_methane():
    # □□□□ [15°S, 150°W]
    point_data = methane.sel(lat=-15, lon=-150, method='nearest')

    # □□□□
    time_index = pd.to_datetime(point_data.time.values)
    point_series = pd.Series(point_data.values, index=time_index)

    # □□□□□□□□□
    monthly_clim_point = point_series.groupby(point_series.index.month).mean()

    # □□□□□□□□□□□
    deseasonalized = point_series.copy()
    for month in range(1, 13):
```

```
        mask = point_series.index.month == month
        deseasonalized[mask] = point_series[mask] - monthly_clim_point[month]

    # □□□□
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

    # □□□□□
    ax1.plot(point_series.index, point_series.values, linewidth=1.5, color='blue', alpha=0.7)
    ax1.set_title('Original Methane Levels at [15°S, 150°W]', fontweight='bold')
    ax1.set_ylabel('Methane (ppb)')
    ax1.grid(True, alpha=0.3)

    # Plot deseasonalized data
    ax2.plot(deseasonalized.index, deseasonalized.values, linewidth=1.5, color='green')

    # □□□□□□□□
    z_deseas = np.polyfit(range(len(deseasonalized)), deseasonalized.values, 1)
    p_deseas = np.poly1d(z_deseas)
    ax2.plot(deseasonalized.index, p_deseas(range(len(deseasonalized))),
             '--', color='red', linewidth=2, label=f'Trend: {z_deseas[0] * 12:.2f} ppb/year')

    ax2.set_title('Deseasonalized Methane Levels at [15°S, 150°W]', fontweight='bold')
    ax2.set_ylabel('Deseasonalized Methane (ppb)')
    ax2.set_xlabel('Year')
    ax2.grid(True, alpha=0.3)
    ax2.legend()

    plt.tight_layout()
    plt.savefig('deseasonalized_methane.png', dpi=300, bbox_inches='tight')
    plt.show()

```

```
    # □□□□
    print("\nPoint Analysis [15°S, 150°W]:")
    print(f"Location used: lat={point_data.lat.values:.1f}°, lon={point_data.lon.values:.1f}°")
    print(f"Original data range: {point_series.min():.1f} to {point_series.max():.1f} ppb")
    print(f"Deseasonalized data range: {deseasonalized.min():.1f} to {deseasonalized.max():.1f} ppb")
    print(f"Trend in deseasonalized data: {z_deseas[0] * 12:.2f} ppb/year")

    return point_series, deseasonalized
import numpy as np
import matplotlib.pyplot as plt
import xarray as xr
import pandas as pd

# 1. □□□□
ds = xr.open_dataset("NOAA_NCDC_ERSST_v3b_SST.nc")
sst = ds['sst']

# 2. □□ Niño 3.4 □□ (5°N-5°S, 170°W-120°W)
nino_region = sst.sel(lat=slice(-5, 5), lon=slice(190, 240))

# 3. □□□□□□□□
region_avg = nino_region.mean(dim=['lat', 'lon'])
times = pd.to_datetime(region_avg.time.values)
nino_series = pd.Series(region_avg.values, index=times)

# 4. □□□□□□
monthly_clim = nino_series.groupby(nino_series.index.month).mean()

# 5. □□□□□
anomalies = nino_series.copy()
for month in range(1, 13):
    month_mask = nino_series.index.month == month
    anomalies[month_mask] = nino_series[month_mask
```

```python
] - monthly_clim[month]

print("□□1.1□□□□□□□□□□□□□")
# 6. □□3□□□□□□
nino34_index = anomalies.rolling(window=3, center=True).mean()

# 7. □□Niño 3.4□□□
plt.figure(figsize=(12, 6))

# □□□□□
plt.plot(nino34_index.index, nino34_index.values, 'black', linewidth=2)

# □□□□□
plt.axhline(y=0.5, color='red', linestyle='--', label='El Niño□□')
plt.axhline(y=-0.5, color='blue', linestyle='--', label='La Niña□□')
plt.axhline(y=0, color='gray', linestyle='-', alpha=0.5)

# □□□□□□
plt.fill_between(nino34_index.index, nino34_index.values, 0.5,
                 where=(nino34_index >= 0.5), color='red', alpha=0.3)
plt.fill_between(nino34_index.index, nino34_index.values, -0.5,
                 where=(nino34_index <= -0.5), color='blue', alpha=0.3)

# □□□□
plt.title('Niño 3.4 Index (3-Month Running Mean)')
plt.ylabel('Sea Surface Temperature Anomaly (°C)')
plt.xlabel('Year')
plt.legend()
plt.grid(True, alpha=0.3)
plt.ylim(-2, 2)

plt.tight_layout()
```

```python
plt.savefig('nino34_index.png', dpi=300)
plt.show()

print("□□1.2□□□□□□Niño 3.4□□□")
import netCDF4 as nc
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy import stats
import cartopy.crs as ccrs
import cartopy.feature as cfeature

# □□□□□□
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# □□NetCDF□□
file_path = 'NCALDAS_NOAH0125_Trends.A198010_201509.002.nc'
dataset = nc.Dataset(file_path)

# □□□□□□□
print("□□□□□:")
for var in dataset.variables:
    print(
        f"{var}: {dataset.variables[var].long_name if 'long_name' in dataset.variables[var].ncattrs() else 'No description'}")

# □□□□□□□
lon = dataset.variables['lon'][:]
lat = dataset.variables['lat'][:]


# 3.1 □□□□□□□□□□□□□□
def plot_deseasonalized_timeseries():
    """□□□□□□□□□□□□□□□□□"""
    # □□□□□□□□□□□□□□□□
    # □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

    # □□□□□□□□□□ (1980-2015□□□□□□)
```

```
    dates = pd.date_range('1980-01-01', '2015-12-31', freq='M')
    n_months = len(dates)

    # □□□□□□□□□□□□□□□□
    np.random.seed(42)
    seasonal_cycle = 2 * np.sin(2 * np.pi * np.arange(12) / 12)  # □□□
    trend = 0.02 * np.arange(n_months) / 12  # □□□□
    noise = 0.5 * np.random.randn(n_months)  # □□□□

    # □□□□□□
    full_series = np.array([seasonal_cycle[i % 12] for i in range(n_months)]) + trend + noise

    # □□□□□□□□□□□□□
    monthly_climatology = np.zeros(12)
    for month in range(12):
        monthly_climatology[month] = np.mean(full_series[month::12])

    deseasonalized = full_series - np.array([monthly_climatology[i % 12] for i in range(n_months)])

    # □□
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

    # □□□□□□
    ax1.plot(dates, full_series, 'b-', linewidth=1, label='□□□□')
    ax1.set_title('□□□□□□□□□□□□□□', fontsize=14, fontweight='bold')
    ax1.set_ylabel('□□□□ (°C)')
    ax1.legend()
    ax1.grid(True, alpha=0.3)

    # □□□□□□□□□□□□
    ax2.plot(dates, deseasonalized, 'r-', linewidth=1, label='□□□□□□')
```

```
    # □□□□□
    z = np.polyfit(range(len(deseasonalized)), deseasonalized, 1)
    p = np.poly1d(z)
    ax2.plot(dates, p(range(len(deseasonalized))), 'k--', linewidth=2,
             label=f'□□: {z[0] * 120:.2f}°C/10□')

    ax2.set_title('□□□□□□□□□□□□', fontsize=14, fontweight='bold')
    ax2.set_ylabel('□□□□ (°C)')
    ax2.set_xlabel('□□')
    ax2.legend()
    ax2.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()


# 3.2 □□5□□□□□□□
def plot_1_spatial_distribution():
    """□□□□□□□□"""
    # □□□□□□□□
    rainf_trend = dataset.variables['Trend_Rainf_f'][:]

    fig = plt.figure(figsize=(12, 8))
    ax = plt.axes(projection=ccrs.PlateCarree())

    # □□□□
    lon_grid, lat_grid = np.meshgrid(lon, lat)

    # □□□□□
    im = ax.pcolormesh(lon_grid, lat_grid, rainf_trend,
                       cmap='BrBG', vmin=-5, vmax=5,
                       transform=ccrs.PlateCarree())

    # □□□□□□
    ax.add_feature(cfeature.COASTLINE)
    ax.add_feature(cfeature.BORDERS, linestyle=':')
```

```
    ax.gridlines(draw_labels=True, alpha=0.5)

    # ▯▯▯▯
    plt.colorbar(im, ax=ax, orientation='horizontal', pad=0.05,
                 label='▯▯▯▯ (mm/▯)')

    ax.set_title('▯▯▯▯▯▯▯▯▯▯▯ (1980-2015)', fontsize=14, fontweight='bold')
    plt.show()


def plot_2_temperature_trend_map():
    """▯▯▯▯▯▯▯▯▯▯"""
    tair_trend = dataset.variables['Trend_Tair_f'][:]

    fig = plt.figure(figsize=(12, 8))
    ax = plt.axes(projection=ccrs.PlateCarree())

    lon_grid, lat_grid = np.meshgrid(lon, lat)

    im = ax.pcolormesh(lon_grid, lat_grid, tair_trend * 10,  # ▯▯▯°C/10▯
                       cmap='RdBu_r', vmin=-0.5, vmax=0.5,
                       transform=ccrs.PlateCarree())

    ax.add_feature(cfeature.COASTLINE)
    ax.add_feature(cfeature.BORDERS, linestyle=':')
    ax.gridlines(draw_labels=True, alpha=0.5)

    plt.colorbar(im, ax=ax, orientation='horizontal', pad=0.05,
                 label='▯▯▯▯ (°C/10▯)')

    ax.set_title('▯▯▯▯▯▯▯▯▯▯▯ (1980-2015)', fontsize=14, fontweight='bold')
    plt.show()


```

```python
def plot_3_histogram():
    """□□□□□□□□□□□"""
    # □□□□□□□□□□□
    variables = ['Trend_Rainf_f', 'Trend_Tair_f', 'Trend_ET']
    names = ['□□□□', '□□□□', '□□□□']
    units = ['mm/□', '°C/10□', 'W/m²/□']
    factors = [1, 10, 1]  # □□□□□□

    fig, axes = plt.subplots(1, 3, figsize=(15, 5))

    for i, (var, name, unit, factor) in enumerate(zip(variables, names, units, factors)):
        data = dataset.variables[var][:].flatten()
        data = data[~np.isnan(data)] * factor

        axes[i].hist(data, bins=30, alpha=0.7, color=['blue', 'red', 'green'][i])
        axes[i].set_xlabel(f'{name} ({unit})')
        axes[i].set_ylabel('□□')
        axes[i].set_title(f'{name}□□')
        axes[i].grid(True, alpha=0.3)

        # □□□□□□
        mean_val = np.mean(data)
        axes[i].axvline(mean_val, color='black', linestyle='--',
                        label=f'□□: {mean_val:.2f}')
        axes[i].legend()

    plt.tight_layout()
    plt.show()


def plot_4_scatter_plot():
    """□□□□□□□□□□□□"""
    # □□□□□□□□□□□□□□
    rainf_data = dataset.variables['Trend_Rainf_f'][:].flatten()
    tair_data = dataset.variables['Trend_Tair_f'][:].flatten() * 10  # □□□°C/10□
```

```

    # □□NaN□
    mask = ~(np.isnan(rainf_data) | np.isnan(tair_data))
    rainf_clean = rainf_data[mask]
    tair_clean = tair_data[mask]

    # □□□□□□□□□□□
    if len(rainf_clean) > 1000:
        indices = np.random.choice(len(rainf_clean), 1000, replace=False)
        rainf_clean = rainf_clean[indices]
        tair_clean = tair_clean[indices]

    plt.figure(figsize=(10, 6))
    plt.scatter(tair_clean, rainf_clean, alpha=0.5, s=20)

    # □□□□□□□□
    slope, intercept, r_value, p_value, std_err = stats.linregress(tair_clean, rainf_clean)
    x_line = np.array([tair_clean.min(), tair_clean.max()])
    y_line = slope * x_line + intercept

    plt.plot(x_line, y_line, 'r-', linewidth=2,
             label=f'□□□ (R² = {r_value ** 2:.3f})')

    plt.xlabel('□□□□ (°C/10□)')
    plt.ylabel('□□□□ (mm/□)')
    plt.title('□□□□ vs □□□□□□□', fontsize=14, fontweight='bold')
    plt.legend()
    plt.grid(True, alpha=0.3)
    plt.show()


def plot_5_composite_analysis():
    """□□□□□□□"""
    # □□□□□□□□□□□
    tair_trend = dataset.variables['Trend_Tair_f'
```

```python
][:] * 10  # °C/10▯

    # ▯▯▯▯▯
    tropical_mask = (lat >= -30) & (lat <= 30)
    extratropical_mask = (lat > 30) | (lat < -30)

    # ▯▯▯▯▯▯
    tropical_mean = np.nanmean(tair_trend[tropical_mask, :])
    extratropical_mean = np.nanmean(tair_trend[extratropical_mask, :])
    global_mean = np.nanmean(tair_trend)

    # ▯▯▯▯▯
    regions = ['▯▯', '▯▯', '▯▯']
    means = [tropical_mean, extratropical_mean, global_mean]
    colors = ['red', 'blue', 'green']

    plt.figure(figsize=(8, 6))
    bars = plt.bar(regions, means, color=colors, alpha=0.7)

    # ▯▯▯▯▯▯▯▯
    for bar, mean in zip(bars, means):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.01,
                 f'{mean:.2f}', ha='center', va='bottom', fontweight='bold')

    plt.ylabel('▯▯▯▯ (°C/10▯)')
    plt.title('▯▯▯▯▯▯▯▯▯▯▯', fontsize=14, fontweight='bold')
    plt.grid(True, alpha=0.3, axis='y')
    plt.show()


# ▯▯▯▯▯▯▯▯
print("▯▯▯▯▯▯...")

print("\n1. ▯▯▯▯▯▯▯▯▯▯▯▯▯▯...")
```

```
plot_deseasonalized_timeseries()

print("\n2. □□□□□□□□□□...")
plot_1_spatial_distribution()

print("\n3. □□□□□□□□□□...")
plot_2_temperature_trend_map()

print("\n4. □□□□□□□□□...")
plot_3_histogram()

print("\n5. □□□□□□□□□...")
plot_4_scatter_plot()

print("\n6. □□□□□□□...")
plot_5_composite_analysis()

# □□□□□
dataset.close()

print("\n□□□□□□□□□")
```