Assignment 3

1. Nino 3.4 index

```
In [1]: import numpy as np
        import pandas as pd
        import xarray as xr
        from matplotlib import pyplot as plt
        %matplotlib inline
```

1.1 compute monthly climatology for SST from Niño 3.4 region, and subtract climatology from SST time series to obtain anomalies.

```
In [2]: # open dataset
        ds = xr.open_dataset("NOAA_NCDC_ERSST_v3b_SST.nc")
        # select nino 3.4 region
        ds_region = ds.sel(lat=slice(-5,5), lon=slice(190,240))
```

Open dataset and select Nino 3.4 region

```
In [3]: #group data by month
        group_data = ds_region.sst.groupby('time.month')

        # Apply mean to grouped data, and then compute the anomalies
        sst_anom = group_data - group_data.mean(dim='time')

        #obtain anomalies result
        print(sst_anom.values)
```

```
[[[-0.43157768 -0.41846275 -0.39795303 ... -0.2116642  -0.23776245
   -0.24401474]
  [-0.41259003 -0.4067192  -0.3875141  ... -0.52064896 -0.5346451
   -0.51997185]
  [-0.40932274 -0.39743805 -0.36237717 ... -0.6373882  -0.6171951
   -0.583725  ]
  [-0.4140854  -0.37909317 -0.3215618  ... -0.43292618 -0.38404274
   -0.3352623 ]
  [-0.5043678  -0.43894005 -0.3710251  ... -0.17453575 -0.11044502
   -0.06918144]]

 [[-0.5374584  -0.52739716 -0.50823593 ... -0.40254593 -0.44382668
   -0.45287704]
  [-0.55093956 -0.539135   -0.51673317 ... -0.6660595  -0.7127285
   -0.710968  ]
  [-0.61242104 -0.5959244  -0.5572338  ... -0.7235069  -0.7326374
   -0.73106194]
  [-0.6798363  -0.6483364  -0.5889931  ... -0.5397434  -0.50793266
   -0.49977684]
  [-0.7830448  -0.7286701  -0.6683655  ... -0.33967972 -0.29167747
```

Group data by month then we can get monthly SST;
(grouped data) – (mean grouped data) = (anomaly SST) then we can get anomaly SST
Print SST anomaly values result above.

1.2 Visualize the computed Niño 3.4. Your plot should look similar to this one.

```
In [4]: # Compute rolling means
        ds_anom_rolling = sst_anom.rolling(time=3, center=True).mean()

        # let anom_rolling 3D to 1D
        line_anom = np.nanmean(ds_anom_rolling.values,axis=(1,2))
```

Compute rolling means (time = 3 monthes);
Rolling mean is a 3D variable we computed above, so we change it to 1D use *np.nanmean()* function. *Nanmean()* means calculate mean values not including Nan values. *axis=(1,2)* means which dimensions will be averaged, in this case: 1 = lat, 2 = lon.

In [5]:
```
# plot
# set time serious
time = pd.date_range(start='1960-01',periods=684,freq='m')
fig,ax = plt.subplots(1,1,figsize = [10,6],dpi=300)

# put time and anomaly into the figure
ax.plot(time,line_anom,color='k')

# set xlabel, ylabel and title
ax.set_ylabel('Anomaly Degrees C', color='k', fontsize=15)
ax.set_xlabel('Year', color='k', fontsize=15)
ax.set_title("SST Anomaly in Niño 3.4 Region", fontsize=20)

# Plot grid lines
ax.grid(linestyle='--',linewidth=0.3,alpha=0.5,color='k')

# put hlines into the figure
ax.hlines(y = 0.5,xmin=time[0],xmax=time[-1],color='r',linestyles='--',lw=0.5,label='EI Nino Threshold')
ax.hlines(y = -0.5,xmin=time[0],xmax=time[-1],color='b',linestyles='--',lw=0.5,label='La Nina Threshold')
ax.hlines(y = 0,xmin=time[0],xmax=time[-1],color='k',linestyles='solid',lw=1,label='3 mth running mean')

# set ylabel limitation
ax.set_ylim(-3,3)

# put legend into the figure
ax.legend(loc='best',fontsize=12)

# fill different color into the figure
ax.fill_between(time,0,line_anom,where=(line_anom>0),color='r')
ax.fill_between(time,0,line_anom,where=(line_anom<0),color='b')
```

Now we plot the picture, and plot the image as similar as possible to the image given by Professor Zhu. Actually, time serious of image from Professor Zhu are different in this case, so I select full time serious from 1960 to 2016.

Create a subplots;

Put variables into the plot (x axis = time, y axis = anomaly);

Following are some detailed operations:

Set ylabel, xlabel and title;

Plot grid lines;

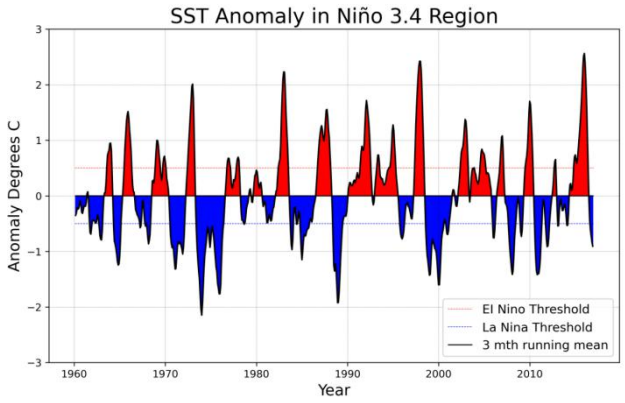
Put highlines into the figure;

Set ylabel limitation;

Put legend into the figure;

Fill different color into the figure.

Finally, the figure output as below:



2. Earth's energy budget

```
In [6]: import numpy as np
        import pandas as pd
        import xarray as xr
        from matplotlib import pyplot as plt
        import nc_time_axis
        %matplotlib inline
```

2.1 Make a 2D plot of the time-mean TOA longwave, shortwave, and solar radiation for all-sky conditions. Add up the three variables above and verify (visually) that they are equivalent to the TOA net flux.

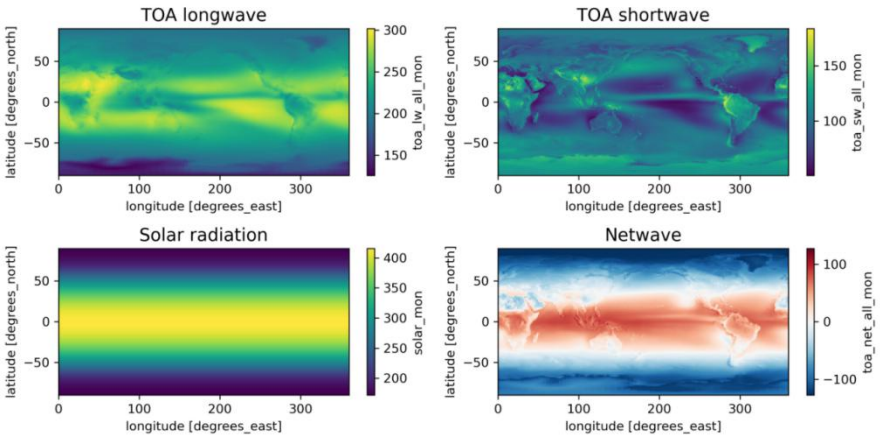
```
In [7]: # open dataset
        ds = xr.open_dataset("CERES_EBAF-TOA_200003-201701.nc")
```

Open dataset.

```
In [8]: # dataarray of TOA longwave, shortwave, solar radiation and netwave
        dalw = ds.toa_lw_all_mon
        dasw = ds.toa_sw_all_mon
        dasolar = ds.solar_mon
        danet = ds.toa_net_all_mon

        # TOA longwave, shortwave, solar radiation and netwave PLOT (mean time)
        fig, axes = plt.subplots(2,2, figsize=(10,5), sharex=False, sharey=False, dpi=300)
        dalw.mean('time').plot(ax=axes[0,0])
        dasw.mean('time').plot(ax=axes[0,1])
        dasolar.mean('time').plot(ax=axes[1,0])
        danet.mean('time').plot(ax=axes[1,1])

        axes[0,0].set_title('TOA longwave',fontsize = 14)
        axes[0,1].set_title('TOA shortwave',fontsize = 14)
        axes[1,0].set_title('Solar radiation',fontsize = 14)
        axes[1,1].set_title('Netwave',fontsize = 14)

        # better layout
        plt.tight_layout()
```

Create 4 different dataarray including TOA longwave, TOA shortwave, solar radiation and netwave.

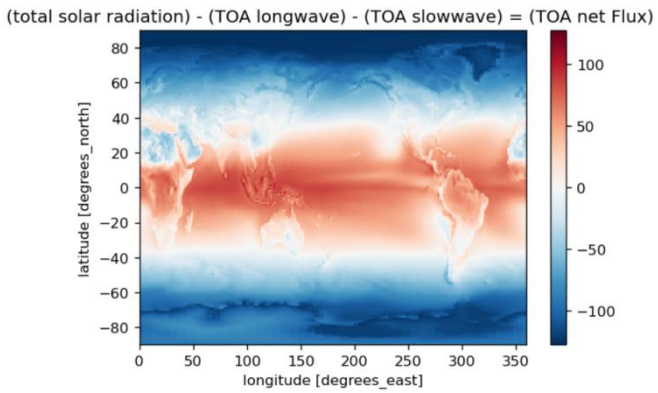
Create 4 subplots and plot 4 different picture into subplots including TOA longwave, TOA shortwave, solar radiation and netwave. Then set title and better layout. Figure shows below:




```
In [9]: # (total solar radiation) - (TOA longwave) - (TOA slowwave) = (TOA net Flux)
        plt.rcParams['figure.dpi'] = 120
        da_total = dasolar - dalw - dasw
        da_total.mean('time').plot()
        plt.title('(total solar radiation) - (TOA longwave) - (TOA slowwave) = (TOA net Flux) ')
        # It is the same as the TOA net plot directly above
```

(total solar radiation) - (TOA longwave) - (TOA shortwave) = (TOA net Flux)

According to the above formula, we operate on the data, then plot the figure blow:

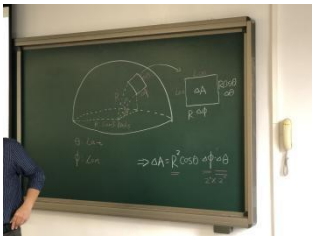


2.2 Calculate and verify that the TOA incoming solar, outgoing longwave, and outgoing shortwave approximately match up with the cartoon above.

In [23]:
```
# Create the weights
weights = np.cos(np.deg2rad(ds.lat))
```

In [27]:
```
weights = np.cos(np.deg2rad(ds.lat))
toa_weighted_solar = dasolar.weighted(weights)
toa_weighted_lw = dalw.weighted(weights)
toa_weighted_sw = dasw.weighted(weights)


print('solar radiations:',toa_weighted_solar.mean(dim=('lon', 'lat','time')).values,'(Wm-2)')
print('long wave outgoing:',toa_weighted_lw.mean(dim=('lon', 'lat','time')).values,'(Wm-2)')
print('short wave outgoing:',toa_weighted_sw.mean(dim=('lon', 'lat','time')).values,'(Wm-2)')
```

```
solar radiations: 340.28326598091286 (Wm-2)
long wave outgoing: 240.2669337478465 (Wm-2)
short wave outgoing: 99.13805276923081 (Wm-2)
```



For this question, I referred to what Professor Zhu taught;
Calculate the weighted average for solar radiation, TOA longwave and TOA shortwave;
Print out the values of solar radiation, long-wave radiation, and short-wave radiation.

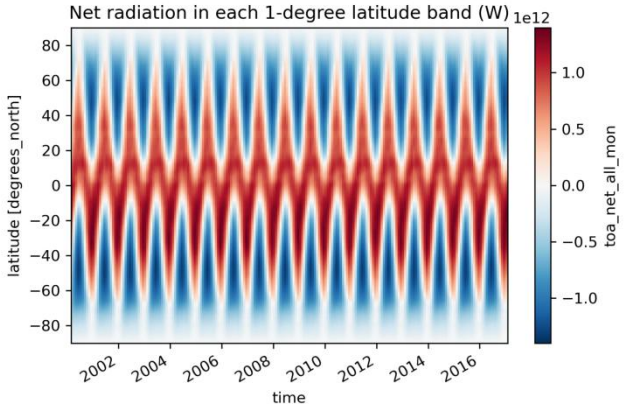
2.3 Calculate and plot the total amount of net radiation in each 1-degree latitude band. Label with correct units.

In [15]:
```
# weight to correct values
danet.values = danet.values * S_lat_lon_time
```

Weight netwave to correct values.

In [16]:
```
# plot the correct netTOA and correct units(W)
danet.mean(dim='lon').transpose().plot()
# add title and better layout
plt.title('Net radiation in each 1-degree latitude band (W)')
plt.tight_layout()
```

Plot the correct TOA net and correct units (W), put title and better layout.
Output figure show below:



2.4 Calculate and plot composites of time-mean outgoing shortwave and longwave radiation for low and high cloud area regions. Here we define low cloud area as ≤25% and high cloud area as ≥75%. Your results should be 2D maps.

In [16]:
```
# open dataset and dataarray of TOA longwave, shortwave
ds = xr.open_dataset("CERES_EBAF-TOA_200003-201701.nc")
dalw = ds.toa_lw_all_mon
dasw = ds.toa_sw_all_mon

# Choose the area of low cloud area and high cloud area
daclda = ds.cldarea_total_daynight_mon
arrclda = daclda.mean(dim='time').values
high_cloud_area = (arrclda>=75)
low_cloud_area = (arrclda<=25)
```

Open dataset and dataarray of TOA longwave, shortwave;
Choose the area of low cloud area and high cloud area.

In [17]:
```
# Calculate highcloud longwave, highcloud shortwave, lowcloud longwave, lowcloud shortwave
hclw=dalw.mean(dim='time').values
hclw[~high_cloud_area]=np.nan
hcsw=dasw.mean(dim='time').values
hcsw[~high_cloud_area]=np.nan
lclw=dalw.mean(dim='time').values
lclw[~low_cloud_area]=np.nan
lcsw=dasw.mean(dim='time').values
lcsw[~low_cloud_area]=np.nan
```

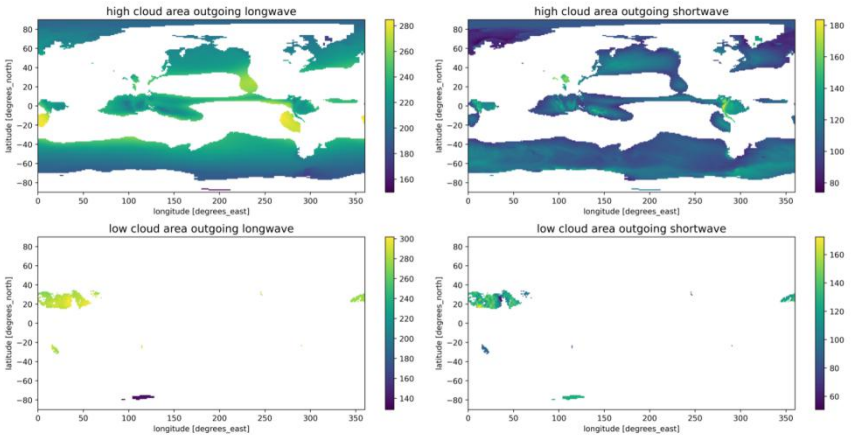
Calculate high cloud longwave, high cloud shortwave, low cloud longwave and low cloud shortwave;
If the selected area is not in the range of high cloud area or low cloud area, then the value is Nan.
Now, hclw, hcsw, lclw and lcsw are array, we need to change it to dataarray.

In [18]:
```
# Introduce dimensions (latitude and longitude) to the 4 arrays calculated above
lat=ds['lat']
lon=ds['lon']
da_hclw = xr.DataArray(hclw,coords=[lat,lon],dims=['lat','lon'])
da_hcsw = xr.DataArray(hcsw,coords=[lat,lon],dims=['lat','lon'])
da_lclw = xr.DataArray(lclw,coords=[lat,lon],dims=['lat','lon'])
da_lcsw = xr.DataArray(lcsw,coords=[lat,lon],dims=['lat','lon'])
```

Introduce dimensions (latitude and longitude) to the 4 arrays calculated above;
Now da_hclw, da_hcsw, da_lclw and da_lcsw are dataarray.

In [19]:
```
# Create Figure and Subplots
fig,axes = plt.subplots(2,2,figsize=(16,8),dpi=300)
# Plot each axes
da_hclw.plot(ax=axes[0,0])
da_hcsw.plot(ax=axes[0,1])
da_lclw.plot(ax=axes[1,0])
da_lcsw.plot(ax=axes[1,1])
axes[0,0].set_title('high cloud area outgoing longwave',fontsize = 14)
axes[0,1].set_title('high cloud area outgoing shortwave',fontsize = 14)
axes[1,0].set_title('low cloud area outgoing longwave',fontsize = 14)
axes[1,1].set_title('low cloud area outgoing shortwave',fontsize = 14)
plt.tight_layout()
```

Cteate figure and subplots;
Plot da_hclw, da_hcsw, da_lclw and da_lcsw in each axes;
Set title for each subplots and better layout. Figures are show below:



2.5 Calculate the global mean values of shortwave and longwave radiation, composited in high and low cloud regions. What is the overall effect of clouds on shortwave and longwave radiation?

In [20]:
```
# Calculate the global mean values of shortwave and longwave radiation
print('high cloud long wave:',np.nanmean(hclw),'(W/m2)')
print('high cloud short wave:',np.nanmean(hcsw),'(W/m2)')
print('low cloud long wave:',np.nanmean(lclw),'(W/m2)')
print('low cloud short wave:',np.nanmean(lcsw),'(W/m2)')
```

```
high cloud long wave: 216.55675 (W/m2)
high cloud short wave: 108.09777 (W/m2)
low cloud long wave: 270.10367 (W/m2)
low cloud short wave: 122.65546 (W/m2)
```

Calculate the global mean values of shortwave and longwave radiation.

3. Explore a netCDF dataset

In [21]:
```
import numpy as np
import pandas as pd
import xarray as xr
from matplotlib import pyplot as plt
import nc_time_axis
%matplotlib inline
```

3.1 Plot a time series of a certain variable with monthly seasonal cycle removed.

In [22]:
```
# open dataset
ds = xr.open_dataset("air.sig995.2012.nc")
```

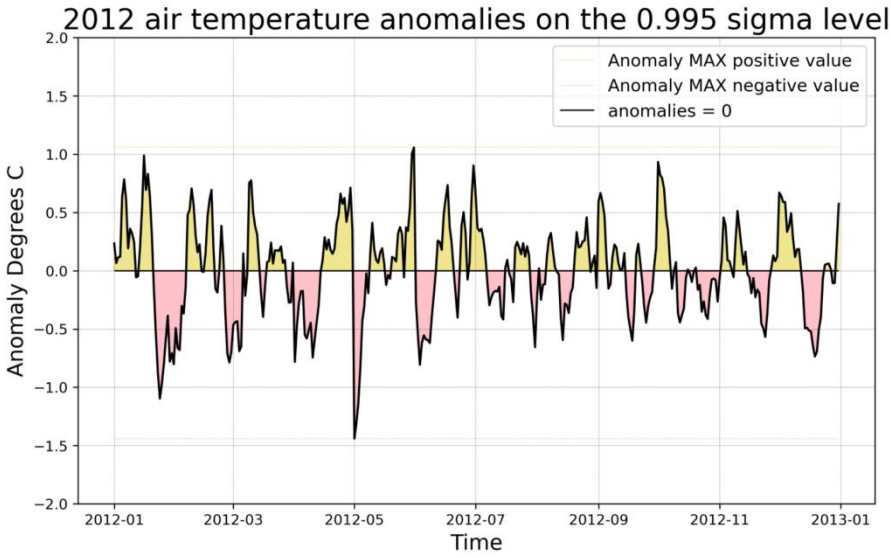
I found a dataset of global average temperature in 2012 and open it.

In [23]:
```
# calculate 2012 air temperature anomalies
group_data = ds.air.groupby('time.month')
air_anom = group_data - group_data.mean(dim='time')
line_air_anom = air_anom.mean(dim={'lat','lon'})
```

Group data by month then we can get monthly air;
(grouped data) – (mean grouped data) = (anomaly air) then we can get anomaly air Temperature.

In [32]:
```
# plot
# set time serious
time = pd.date_range(start='2012-01-01',periods=366,freq='d')
fig,ax = plt.subplots(1,1,figsize = [10,6],dpi=300)

# put time and anomaly into the figure
ax.plot(time,line_air_anom,color='k')

# set xlabel, ylabel and title
ax.set_ylabel('Anomaly Degrees C', color='k', fontsize=15)
ax.set_xlabel('Time', color='k', fontsize=15)
ax.set_title("2012 air temperature anomalies on the 0.995 sigma level", fontsize=20)

# Plot grid lines
ax.grid(linestyle='--',linewidth=0.3,alpha=0.5,color='k')

# put hlines into the figure
ax.hlines(y = line_air_anom.max(),xmin=time[0],xmax=time[-1],color='khaki',linestyles='--',lw=0.5,label='Anomaly MAX pc
ax.hlines(y = line_air_anom.min(),xmin=time[0],xmax=time[-1],color='pink',linestyles='--',lw=0.5,label='Anomaly MAX ne
ax.hlines(y = 0,xmin=time[0],xmax=time[-1],color='k',linestyles='solid',lw=1,label='anomalies = 0')

# set ylabel limitation
ax.set_ylim(-2,2)
# put legend into the figure
ax.legend(loc='best',fontsize=12)

# fill different color into the figure
ax.fill_between(time,0,line_air_anom,where=(line_air_anom>0),color='khaki')
ax.fill_between(time,0,line_air_anom,where=(line_air_anom<0),color='pink')
```

Set time serious, start from 2012-01-01, periods are 366 days and frequent is day;
Cteate figure and subplots;
Put variables into the plot (x axis = time, y axis = anomaly);
Following are some detailed operations:
Set ylabel, xlabel and title;
Plot grid lines;
Put highlines into the figure;
Set ylabel limitation;
Put legend into the figure;
Fill different color into the figure.
Finally, the figure output as below:



3.2 Make at least 5 different plots using the dataset.

```
In [33]: ds = xr.open_dataset("air.sig995.2012.nc")
```

Open dataset.

```
In [34]: fig, axes = plt.subplots(2,3, figsize=(12,6), sharex=False, sharey=False, dpi=300)
         da_air = ds.air
         da_air_Dec = ds.air.sel(time=slice('2012-12-01', '2012-12-31'))
         da_air_Jul = ds.air.sel(time=slice('2012-07-01','2012-07-31'))
         da_air_shenzhen = ds.air.sel(lon='114',lat='22.5',method='nearest')
         da_air_Ant = ds.air.sel(lat='-90',method='nearest')



         da_air_Dec.mean('time').plot(ax=axes[0,0])
         da_air_Jul.mean('time').plot(ax=axes[1,0],cmap='rainbow')
         da_air_Dec.mean('lon').transpose().plot(ax=axes[0,1])
         da_air_Jul.mean('lon').transpose().plot(ax=axes[1,1],cmap='rainbow')
         da_air_shenzhen.plot(ax=axes[1,2],c='r')
         da_air_Ant.mean('lon').plot(ax=axes[0,2])


         axes[0,2].grid(linestyle='--', linewidth=0.5, alpha=0.5)
         axes[1,2].grid(linestyle='--', linewidth=0.5, alpha=0.5)


         axes[0,0].set_title('Mean temperature in Dec 2012 (K)',fontsize = 14)
         axes[1,0].set_title('Mean temperature in Jul 2012 (K)',fontsize = 14)
         axes[0,1].set_title('Temperature in Dec 2012 mean lon (K)',fontsize = 14)
         axes[1,1].set_title('Temperature in Jul 2012 mean lon (K)',fontsize = 14)
         axes[1,2].set_title('Mean temperature in ShenZhen 2012 (K)',fontsize = 14)
         axes[0,2].set_title('Mean temperature in Antarctica 2012 (K)',fontsize = 14)

         # better layout
         plt.tight_layout()
```

Cteate figure and subplots;

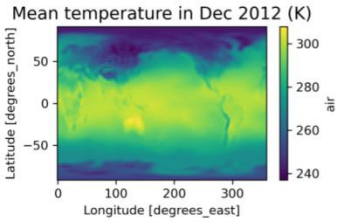
The 6 pictures I want to show are:

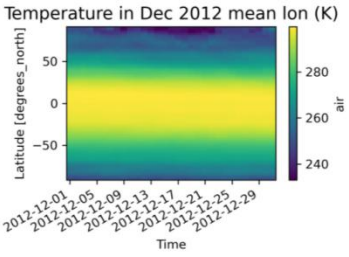
1. December global average temperature
2. Global average temperature in June
3. Average temperature at different latitudes in December
4. Average temperature at different latitudes in June
5. Shenzhen's annual temperature change curve
6. Annual temperature change curve at South Pole

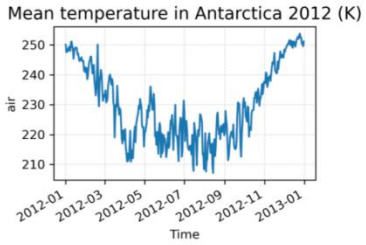
```
da_air_Dec = ds.air.sel(time=slice('2012-12-01', '2012-12-31'))
da_air_Jul = ds.air.sel(time=slice('2012-07-01','2012-07-31'))
da_air_shenzhen = ds.air.sel(lon='114',lat='22.5',method='nearest')
da_air_Ant = ds.air.sel(lat='-90',method='nearest')
```

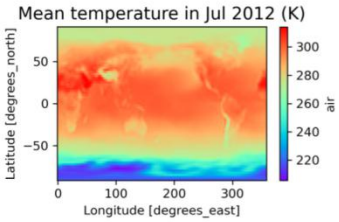
So first of all need to select the time period (*slice()*), and select (*lon and lat*)

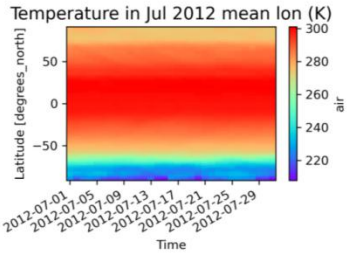
Set title for each subplots and better layout. Figures are show below:

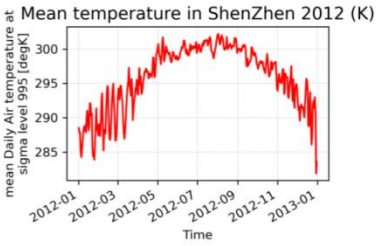

Mean temperature in Dec 2012 (K)
Latitude [degrees_north]
50
0
−50
0
100
200
300
Longitude [degrees_east]
300
280
260
240
air


Temperature in Dec 2012 mean lon (K)
Latitude [degrees_north]
50
0
−50
2012-12-01
2012-12-05
2012-12-09
2012-12-13
2012-12-17
2012-12-21
2012-12-25
2012-12-29
Time
280
260
240
air


Mean temperature in Antarctica 2012 (K)
air
250
240
230
220
210
2012-01
2012-03
2012-05
2012-07
2012-09
2012-11
2013-01
Time


Mean temperature in Jul 2012 (K)
Latitude [degrees_north]
50
0
−50
0
100
200
300
Longitude [degrees_east]
300
280
260
240
220
air


Temperature in Jul 2012 mean lon (K)
Latitude [degrees_north]
50
0
−50
2012-07-01
2012-07-05
2012-07-09
2012-07-13
2012-07-17
2012-07-21
2012-07-25
2012-07-29
Time
300
280
260
240
220
air


Mean temperature in ShenZhen 2012 (K)
mean Daily Air temperature at sigma level 995 [degK]
300
295
290
285
2012-01
2012-03
2012-05
2012-07
2012-09
2012-11
2013-01
Time