```
import numpy as np
from astropy.io import fits
import sep
from astropy.visualization import ZScaleInterval,ImageNormalize
from mpl_toolkits.axes_grid1 import make_axes_locatable
import matplotlib.pyplot as plt

# Here we will write a utility that let's us play with sky subtraction para
# single image, and display the before and after result.  In the other word
# is to write a function skysubtract(filename,F,B) that uses the sky parame
# to subtract the sky, and make plots to display the before/after.

# skySubtract(filename,F=3,B=64,savename= "skysubtract.png")


def skySubtract(filename,F,B,savename=None):
    # Open the file and read the image data
    hdu = fits.open(filename) # Open the file
    image_data = hdu[0].data # get the data (if they're calibrated,
    # they're already floats)

    # This line is need for SEP; if you get a byteswap error, comment this
    image_data = image_data.byteswap(inplace=True).newbyteorder()

    # use SEP to determine the background sky level
    sky = sep.Background(image_data,fw=F,fh=F,bh=B,bw=B) # Background Sky L

    # This is the 2D array containing the sky level in each pixel (ADUs)
    ## sky_data = sky.back(filename)
    sky_data = sky.back()

    print(f"Average sky level in ADUs: {np.mean(sky_data):.2f}")
    ############ DEBUG

    # Print statistics
    print(f"Average sky level in ADUs (before subtraction):\
          {np.mean(sky_data):.2f}")
    print("Min Image:", np.min(image_data))
    print("Max Image:", np.max(image_data))
    print("Mean Image:", np.mean(image_data))
    print("Median Image:", np.median(image_data))
    print("Data Type:", image_data.dtype)
```

```
    ################
    # Subtract the sky data from the image data
    image_data_nosky = image_data - sky_data

    # Call the function makeplots defined below
    makeplots(image_data,sky_data,image_data_nosky,F,B,savename=savename)



def makeplots(image_data,sky_data,image_data_nosky,F,B,savename=None):
    # This function is complete.  No need to edit.
    # Plot the 3 images side by side to compare them (the original image,
    # the sky image, and the image minus the sky)

    fig, ax = plt.subplots(1,3,figsize=[12,3])

    # scales the image by same 'zscale' algorithm as ds9
    norm = ImageNormalize(image_data,interval=ZScaleInterval())
    im0 = ax[0].imshow(image_data,origin='lower',cmap='gray',norm=norm)
    ax[0].set_title('Original')

    im1 = ax[1].imshow(sky_data,origin='lower',cmap='gray') # linear.
    # no need for zscale
    ax[1].set_title(f'Sky; F={F}, B={B}')

    # scales the image by same 'zscale' algorithm as ds9
    norm = ImageNormalize(image_data_nosky,interval=ZScaleInterval())
    im2 = ax[2].imshow(image_data_nosky,origin='lower',cmap='gray',norm=nor
    ax[2].set_title('After sky subtraction')

    # Remove the ticks and tick labels
    for a in ax:
        a.xaxis.set_visible(False)
        a.yaxis.set_visible(False)

    # Add colour bars to all three panels (not as simple when using subplot
    # calls function below)
    colourbar(im0,ax[0])
    colourbar(im1,ax[1])
    colourbar(im2,ax[2])
    fig.tight_layout()
    if savename: fig.savefig(savename)
```

```python
def colourbar(sc,ax):
    # This function is complete.  No need to edit.
    divider = make_axes_locatable(ax)
    cax = divider.append_axes('right', size="5%", pad=0.05)
    cbar = plt.colorbar(sc, cax=cax, orientation='vertical')
```