```python
from skimage.io import imread
from skimage.data import camera
from scipy.ndimage import map_coordinates
import numpy as np
import matplotlib.pyplot as plt

def linear_polar(img, o=None, r=None, output=None, order=1, cont=0):
    if o is None: o = np.array(img.shape[:2])/2 - 0.5
    if r is None: r = (np.array(img.shape[:2])**2).sum()**0.5/2
    if output is None:
        shp = int(round(r)), int(round(r*2*np.pi))
        output = np.zeros(shp, dtype=img.dtype)
    elif isinstance(output, tuple):
        output = np.zeros(output, dtype=img.dtype)
    out_h, out_w = output.shape
    out_img = np.zeros((out_h, out_w), dtype=img.dtype)
    rs = np.linspace(0, r, out_h)
    ts = np.linspace(0, np.pi*2, out_w)
    xs = rs[:,None] * np.cos(ts) + o[1]
    ys = rs[:,None] * np.sin(ts) + o[0]
    map_coordinates(img, (ys, xs), order=order, output=output)
    return output

def polar_linear(img, o=None, r=None, output=None, order=1, cont=0):
    if r is None: r = img.shape[0]
    if output is None:
        output = np.zeros((r*2, r*2), dtype=img.dtype)
    elif isinstance(output, tuple):
        output = np.zeros(output, dtype=img.dtype)
    if o is None: o = np.array(output.shape)/2 - 0.5
    out_h, out_w = output.shape
    ys, xs = np.mgrid[:out_h, :out_w] - o[:,None,None]
    rs = (ys**2+xs**2)**0.5
   # breakpoint()
    ts = np.arccos(xs/rs)
    ts[ys<0] = np.pi*2 - ts[ys<0]
    ts *= (img.shape[1]-1)/(np.pi*2)
    map_coordinates(img, (rs, ts), order=order, output=output)
    return output

if __name__ == '__main__':
    img = camera()
    ax = plt.subplot(311)
    ax.imshow(img)
    out = linear_polar(img)
    ax = plt.subplot(312)
    ax.imshow(out)
    img = polar_linear(out, output=img.shape)
    ax = plt.subplot(313)
    ax.imshow(img)
    plt.show()
```