범례	코드내용	필수 사항	선택 사항	설명	변경 가능
	색상/서식	code	code	code	**code**

#	코드	주석
0	#!/usr/bin/env python ############################ # GK2A L1B data Processing sample code2 # # This program extracts user defined area's pixel value/latitude/longitude value from GK2A NetCDF4 file # and converts digital count number to Albedo/Brightness Temperature. # After that it saves converted data to new NetCDF4 file with geographic coordinates. # # Input: GK2A L1B file [sample file: SW038/fd020ge] (netCDF4) # GK2A conversion table(ASCII) # # process: read input files -> cut user defined area from input # -> convert digital count number to Albedo/Brightness Temperature # -> save data with netCDF4 form # # Output : Albedo/Brightness Temperature from cut area (netCDF4) # # The output netCDF4 file includes next data # -user defined area's line & column size # -user defined area's image_pixel_values # -latitude & longitude of every pixel in cut area # -user defined area's line/column number of left upper point in original GEOS image array(in global attribute) # -user defined area's line/column number of right lower point in original GEOS image array(in global attribute)	프로그램 헤더부 프로그램 기능 및 입/출력 파일 설명
	############################ # Library import & Function define ############################	라이브러리 준비 및 함수정의구간 시작
1	import netCDF4 import numpy	- 파이썬 라이브러리 사용 준비 단계

2	def latlon_from_lincol_geos(Resolution, Line ,Column):
	degtorad=3.14159265358979 / 180.0
	if(Resolution == 0.5):
	COFF=11000.5
	CFAC=8.170135561335742e7
	LOFF=11000.5
	LFAC=8.170135561335742e7
	elif(Resolution == 1.0):
	COFF=5500.5
	CFAC=4.0850677806678705e7
	LOFF=5500.5
	LFAC=4.0850677806678705e7
	else:
	COFF=2750.5
	CFAC=2.0425338903339352e7
	LOFF=2750.5
	LFAC=2.0425338903339352e7
	sub_lon=128.2
	sub_lon=sub_lon*degtorad
	x= degtorad *((Column - COFF)*2**16 / CFAC)
	y= degtorad *((Line - LOFF)*2**16 / LFAC)
	Sd = np.sqrt((42164.0*np.cos(x)*np.cos(y))**2 - (np.cos(y)**2 + 1.006739501*np.sin(y)**2)*1737122264)
	Sn = (42164.0*np.cos(x)*np.cos(y)-Sd) / (np.cos(y)**2 + 1.006739501*np.sin(y)**2)
	S1 = 42164.0 - (Sn * np.cos(x) * np.cos(y))
	S2 = Sn * (np.sin(x) * np.cos(y))
	S3 = -Sn * np.sin(y)
	Sxy = np.sqrt(((S1*S1)+(S2*S2)))
	nlon=(np.arctan(S2/S1)+sub_lon)/degtorad
	nlat=np.arctan((1.006739501 *S3)/Sxy)/degtorad
	return (nlat, nlon)
		- 함수 정의:
		전구 영역 GEOS 도법 자료의 행과 열 번호를 입력받아 위경도를 출력하는 함수

3

```
def lincol_from_latlon_geos(Resolution, Latitude, Longitude):
        degtorad=3.14159265358979 / 180.0
        if(Resolution == 0.5):
                COFF=11000.5
                CFAC=8.170135561335742e7
                LOFF=11000.5
                LFAC=8.170135561335742e7
        elif(Resolution == 1.0):
                COFF=5500.5
                CFAC=4.0850677806678705e7
                LOFF=5500.5
                LFAC=4.0850677806678705e7
        else:
                COFF=2750.5
                CFAC=2.0425338903339352e7
                LOFF=2750.5
                LFAC=2.0425338903339352e7
                sub_lon=128.2
        sub_lon=sub_lon*degtorad
        Latitude=Latitude*degtorad
        Longitude=Longitude*degtorad
                c_lat = np.arctan(0.993305616*np.tan(Latitude))
        RL =  6356.7523 / np.sqrt( 1.0 - 0.00669438444*np.cos(c_lat)**2.0 )
        R1 =  42164.0 - RL *np.cos(c_lat)*np.cos(Longitude - sub_lon)
        R2 = -RL* np.cos(c_lat) *np.sin(Longitude - sub_lon)
        R3 =  RL* np.sin(c_lat)
        Rn =  np.sqrt(R1**2.0 + R2**2.0 + R3**2.0 )
                x = np.arctan(-R2 / R1) / degtorad
        y = np.arcsin(-R3 / Rn) / degtorad
        ncol=COFF + (x* 2.0**(-16) * CFAC)
        nlin=LOFF + (y* 2.0**(-16) * LFAC)
        return (nlin,ncol)
```

- 함수 정의:
위경도를 입력받고 전구 영역 GEOS 도법 자료의 해당 행과 열의 번호를 출력하는 함수

4	def cut_with_latlon_geos(Array, Resolution, Latitude1, Longitude1, Latitude2, Longitude2):
	Array=np.array(Array)
	if(Resolution == 0.5):
	Index_max=22000
	elif(Resolution == 1.0):
	Index_max=11000
	else:
	Index_max=5500
	(Lin1,Col1) = lincol_from_latlon_geos(Resolution, Latitude1, Longitude1)
	(Lin2,Col2) = lincol_from_latlon_geos(Resolution, Latitude2, Longitude2)
	Col1=int(np.floor(Col1))
	Lin1=int(np.floor(Lin1))
	Col2=int(np.ceil(Col2))
	Lin2=int(np.ceil(Lin2))
	cut=np.zeros((Index_max,Index_max))
	if((Col1 <= Col2) and (Lin1 <= Lin2) and (0 <= Col1) and (Col2 < Index_max) and (0 <= Lin1) and (Lin2 < Index_max)):
	cut=Array[Lin1:Lin2,Col1:Col2]
	return cut
		- 함수 정의 :
		전구 영역 GEOS 도법 자료의 위도, 경도, 픽셀값 행렬을 사용자 정의 영역대로 잘라내는 함수
		
		입력 변수
		-Array: 전구 영역 GEOS 도법 자료의 image_pixel_values/위경도 행렬 [array/numpy array]
		
		-Resolution: 전구 영역 GEOS 도법 자료의 해상도(km) [float]
		
		-Latitude1: 사용자 정의영역의 좌상단에 해당하는 위도 (degree) [float]
		-Longitude1: 사용자 정의영역의 좌상단에 해당하는 경도 (degree) [float]
		
		-Latitude2: 사용자 정의영역의 우하단에 해당하는 위도 (degree) [float]
		-Longitude2: 사용자 정의영역의 우하단에 해당하는 경도 (degree) [float]
		
		출력
		사용자 정의 영역에 해당하는 image_pixel_values/위경도 행렬 [numpy array]
		
		입력 변수는 항상 아래의 부등식을 만족해야한다.
		
		Latitude1 >= Latitude2
		Longitude1 <= Latitude2

	############################ #Main Program Start ############################	샘플 코드 주요 처리영역 시작

5	input_ncfile_path = '**gk2a_ami_le1b_sw038_fd020ge_201905100300.nc**' CT_path='**./conversion_table/**' left_upper_lat=**45.728965** left_upper_lon=**113.996417** right_lower_lat=**29.312252** right_lower_lon=**135.246740** output_ncfile_path='**output_ncfile.nc**'	- 입력 자료 설정 >L1B 자료 파일 이름 >Calibration Table 디렉토리 >사용자정의영역 좌상단, 우하단 위경도 - 출력 자료 설정 >이진(nc)파일 이름
6	input_ncfile = nc.Dataset(input_ncfile_path,'r',format='netcdf4') ipixel=input_ncfile.variables['image_pixel_values']	- image_pixel_values 추출 >netCDF4 파일 불러오기 >image_pixel_values 불러오기 ->ipixel
7	i = np.arange(0,input_ncfile.getncattr('number_of_columns'),dtype='f') j = np.arange(0,input_ncfile.getncattr('number_of_lines'),dtype='f') i,j = np.meshgrid(i,j) (geos_lat,geos_lon) = latlon_from_lincol_geos(2.0,j,i)	- 위경도 계산 전구 영역 GEOS 도법 자료에서 각 픽셀별 위경도를 계산
8	cut_pixel=cut_with_latlon_geos(ipixel[:],2.0,left_upper_lat,left_upper_lon,right_lower_lat,right_lower_lon) cut_lat=cut_with_latlon_geos(geos_lat,2.0,left_upper_lat,left_upper_lon,right_lower_lat,right_lower_lon) cut_lon=cut_with_latlon_geos(geos_lon,2.0,left_upper_lat,left_upper_lon,right_lower_lat,right_lower_lon) (ulc_lin,ulc_col)=lincol_from_latlon_geos(2.0,left_upper_lat,left_upper_lon) (lrc_lin,lrc_col)=lincol_from_latlon_geos(2.0,right_lower_lat,right_lower_lon)	- 사용자 정의 영역 추출 전구 영역 GEOS 도법 자료에서 사용자 정의 영역을 잘라냄.
9	cut_pixel[cut_pixel>49151] = 0 #set error pixel's value to 0	- DQF 처리 Error pixel(Ⅲ-5참조) 값을 0으로 필터링

10	channel=ipixel.getncattr('channel_name')
	if ((channel == 'VI004') or (channel == 'VI005') or (channel == 'NR016')):
	mask = 0b0000011111111111 #11bit mask
	elif ((channel == 'VI006') or (channel == 'NR013') or (channel == 'WV063')):
	mask = 0b0000111111111111 #12bit mask
	elif (channel == 'SW038'):
	mask = 0b0011111111111111 #14bit mask
	else:
	mask = 0b0001111111111111 #13bit mask
	cut_pixel_masked=np.bitwise_and(cut_pixel,mask)
		- Bit Masking
		> 채널별 Bit size per pixel(Ⅲ-6참조)에 따라 bit mask 생성 및 마스크 처리

11	AL_postfix='_con_alb.txt'
	BT_postfix='_con_bt.txt'
	if (channel[0:2] == 'VI') or (channel[0:2] == 'NR'):
	conversion_table=np.loadtxt(CT_path+channel+AL_postfix,'float64')
	convert_data='albedo'
	else:
	conversion_table=np.loadtxt(CT_path+channel+BT_postfix,'float64')
	convert_data='brightness_temperature'
	cut_pixel_masked_converted=conversion_table[cut_pixel_masked] # pixel data : table value / 1:1 matching
	input_ncfile.close()
		- Albedo/BT 변환
		> 픽셀 값을 알베도/밝기온도로 변환
		(IV-5참조)

12	output_ncfile=nc.Dataset(output_ncfile_path,'w',format='NETCDF4') data_lin_max=output_ncfile.createDimension("data_lin_max",cut_pixel.shape[0]) data_col_max=output_ncfile.createDimension("data_col_max",cut_pixel.shape[1]) output_ncfile.createVariable(convert_data,np.float32,("data_lin_max","data_col_max",)) output_ncfile.createVariable('latitude',np.float64,("data_lin_max","data_col_max",)) output_ncfile.createVariable('longitude',np.float64,("data_lin_max","data_col_max",)) output_ncfile.variables[convert_data][:]=cut_pixel_masked_converted output_ncfile.variables['latitude'][:]=cut_lat output_ncfile.variables['longitude'][:]=cut_lon output_ncfile.left_upper_lin_col_from_geos="line number(start from 0):"+str(int(np.floor(ulc_lin)))+" column number(start from 0):"+str(int(np.floor(ulc_col))) output_ncfile.right_lower_lin_col_from_geos="line number(start from 0):"+str(int(np.ceil(lrc_lin)))+" column number(start from 0):"+str(int(np.ceil(lrc_col))) output_ncfile.close()	- NetCDF4 형태로 출력 >output_ncfile.nc
	############################ #End of Program ############################	프로그램 종료