In [1]:
```
import sys
```

In [2]:
```
sys.path.append('<snappy-dir>')
```

In [3]:
```
import snappy
```

In [4]:
```
from snappy import ProductIO
```

In [5]:
```
p= ProductIO.readProduct('C:/Users/sarit/anaconda3/envs/snap/Lib/snappy/testdata/MER_FRS_L1B_S
list(p.getBandNames())
```

Out[5]:
```
['radiance_1',
 'radiance_2',
 'radiance_3',
 'radiance_4',
 'radiance_5',
 'radiance_6',
 'radiance_7',
 'radiance_8',
 'radiance_9',
 'radiance_10',
 'radiance_11',
 'radiance_12',
 'radiance_13',
 'radiance_14',
 'radiance_15',
 'l1_flags',
 'detector_index']
```

In [6]:
```
import os
snappy_envar = 'USERPROFILE'
envs = os.environ
if not snappy_envar in envs.keys():
    raise Exception('Can’t find snappy')
else:
    snappy_dir = os.path.join(envs.get(snappy_envar), '.snap', 'snap-python')
sys.path.append(snappy_dir)
import snappy
```

In [7]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\sarit\anaconda3\envs\snap\lib\site-packages
(1.19.5)
```

In [8]:
```
!pip install matplotlib.colors
```

```
Requirement already satisfied: matplotlib.colors in c:\users\sarit\anaconda3\envs\snap\lib\s
ite-packages (1.0.16)
Requirement already satisfied: matplotlib in c:\users\sarit\anaconda3\envs\snap\lib\site-pac
kages (from matplotlib.colors) (3.3.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\sarit\an
aconda3\envs\snap\lib\site-packages (from matplotlib->matplotlib.colors) (3.0.4)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\sarit\anaconda3\envs\snap\li
b\site-packages (from matplotlib->matplotlib.colors) (2.8.2)
Requirement already satisfied: numpy>=1.15 in c:\users\sarit\anaconda3\envs\snap\lib\site-pa
ckages (from matplotlib->matplotlib.colors) (1.19.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sarit\anaconda3\envs\snap\lib\s
ite-packages (from matplotlib->matplotlib.colors) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\sarit\anaconda3\envs\snap\lib\site-p
ackages (from matplotlib->matplotlib.colors) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\sarit\anaconda3\envs\snap\lib\site-
packages (from matplotlib->matplotlib.colors) (8.4.0)
Requirement already satisfied: six>=1.5 in c:\users\sarit\anaconda3\envs\snap\lib\site-packa
ges (from python-dateutil>=2.1->matplotlib->matplotlib.colors) (1.16.0)
```

In [9]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\sarit\anaconda3\envs\snap\lib\site-package
s (1.1.5)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\sarit\anaconda3\envs\snap
\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.15.4 in c:\users\sarit\anaconda3\envs\snap\lib\site-
packages (from pandas) (1.19.5)
Requirement already satisfied: pytz>=2017.2 in c:\users\sarit\anaconda3\envs\snap\lib\site-p
ackages (from pandas) (2023.3.post1)
Requirement already satisfied: six>=1.5 in c:\users\sarit\anaconda3\envs\snap\lib\site-packa
ges (from python-dateutil>=2.7.3->pandas) (1.16.0)
```

In [10]:
```
#1.Load Python Modules
# module                                #  Description
import matplotlib.colors as colors      #  create visualizations
import matplotlib.image as mpimg        #  create visualizations
import matplotlib.pyplot as plt         #  create visualizations
import colorama                         #  prints colored text
import zipfile                          #  zip file manipulation
from os.path import join                #  Data access in file manager
from glob import iglob                  #  Data access in file manager
import pandas as pd                     #  Data analysis and manipulation
import numpy as np                      #  scientific computing
import subprocess                       #  external calls to system
import snappy                           #  SNAP python interface
import jpy                              #  python- Java bridge
# change module setting
pd.options.display.max_colwidth=80      #  longer text in pd.df
```

In [11]:
```
# 2 . User defined functions


```

In [12]:
```
def output_view(product, band, min_value_VV , max_value_VV, min_value_VH , max_value_VH):
    '''
    Creates visualization of processed sentinel-1 SAR data

    Keyword arguments:
    product      --snappy GPF product --> input sentinel-1 product
    band         --list --> product's band to be visualized
    min_value_VV --int  --> min value for color strech in VV band
    max_value_VV --int  --> max value for color strech in VV band
    min_value_VH --int  --> min value for color strech in VH band
    max_value_VH --int  --> max value for color strech in VH band

    '''

    band_data_list = []
    for i in band:
        band = product.getBand(i)
        w = band.getrasterwidth()
        h = band.getRasterHeight()
        band_data = np.zeros(w * h,np.float32)
        band.readPixels(0, 0, w, h, band_data)
        band_data.shape = h, w
        band_data_list.append(band_data)

    fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16 ,16))
    ax1.imshow(band_data_list[0], cmap = 'gary', vmin=min_value_VV , vmax=max_value_VV)
    ax1.set_title(output_bands[0])
    ax2.iashow(band_data_list[1], cmap='gray', vmin=min_value_VH , vamx=max_value_VH)
    ax2.set_title(output_bands[1])

    for ax in fig.get_axes():
        ax.label_outer()
```

In [13]:
```
# call gpt -h from command line
print(subprocess.Popen(['gpt','-h','Subset'] , stdout=subprocess.PIPE, universal_newlines=Tru
```

```
Usage:
  gpt Subset [options]

Description:
  Create a spatial and/or spectral subset of a data product.


Source Options:
  -Ssource=<file>    The source product to create the subset from.
                     This is a mandatory source.

Parameter Options:
  -PcopyMetadata=<boolean>                      Whether to copy the metadata of the source p
roduct.
                                                Default value is 'false'.
  -PfullSwath=<boolean>                         Forces the operator to extend the subset reg
ion to the full swath.
                                                Default value is 'false'.
  -PgeoRegion=<geometry>                        The subset region in geographical coordinate
s using WKT-format,
                                                e.g. POLYGON(({lon1} {lat1}, {lon2} {lat2},
..., {lon1} {lat1}))
                                                (make sure to quote the option due to spaces
in {geometry}).
                                                If not given, the entire scene is used.
  -PreferenceBand=<string>                      The band used to indicate the pixel coordina
tes.
  -Pregion=<rectangle>                          The subset region in pixel coordinates.
                                                Use the following format: {x},{y},{width},{h
eight}
                                                If not given, the entire scene is used. The
'geoRegion' parameter has precedence over this parameter.
  -PsourceBands=<string,string,string,...>      The list of source bands.
  -PsubSamplingX=<int>                          The pixel sub-sampling step in X (horizontal
image direction)
                                                Default value is '1'.
  -PsubSamplingY=<int>                          The pixel sub-sampling step in Y (vertical i
mage direction)
                                                Default value is '1'.
  -PtiePointGrids=<string,string,string,...>    The list of tie-point grid names.

Graph XML Format:
  <graph id="someGraphId">
    <version>1.0</version>
    <node id="someNodeId">
      <operator>Subset</operator>
      <sources>
        <source>${source}</source>
      </sources>
      <parameters>
        <sourceBands>string,string,string,...</sourceBands>
        <tiePointGrids>string,string,string,...</tiePointGrids>
        <region>rectangle</region>
        <referenceBand>string</referenceBand>
        <geoRegion>geometry</geoRegion>
        <subSamplingX>int</subSamplingX>
        <subSamplingY>int</subSamplingY>
        <fullSwath>boolean</fullSwath>
        <copyMetadata>boolean</copyMetadata>
      </parameters>
    </node>
  </graph>
```

In [14]:
```
# set target folder and extract metadata
product_path = "E:\sun\S1A_IW_SLC__1SDV_20230729T010329_20230729T010357_049631_05F7D5_8E7C.zi
#C:\\Users\\sarit\\SAR\\ALPSRS270982150-L1.5\\ALPSRS270982150-L1.5\\VOL-ALPSRS270982150-L1.5_
input_S1_files=sorted(list(iglob(join(product_path,'**' , '*S1* .zip'),recursive=True )))
name,sensing_mode, product_type , polarization, height,width,band_names =([] for i in range(7

for i in input_S1_files:
    sensing_mode.append(i.split("_")[3])
    product_type.append(i.split("_")[4])
    polarization.append(i.split("_")[-6])
    # Read with snappy
    s1_read=snappy.ProductIO.readProduct(i)
    name.append(s1_read.getName())
    height.append(s1_read.getSceneRasterHeight())
    width.append(s1_read.getSceneRasterWidth())
    band_names.append(s1_read.getBandNames())

df_s1_read=pd.DataFrame({'Name': name, 'Sensing Mode': sensing_mode, 'Product Type': product_
                         )
display(df_s1_read)

#display quicklook - first image
with zipfile(input_S1_files[0], 'r') as qck_look:
    qck_look=qck_look.open(name[0]+'.SAFE/preview/quick-look.png')
    img=mpimg.imread(qck_look)
    plt.figure(figsize=(15,15))
    plt.title('Quicklook visualization - '+ name[0]+'\n')
    plt.axis('off')
    plt.imshow(img);
```

| | Name | Sensing Mode | Product Type | Polarization | Height | Width | Band_names |
|---|---|---|---|---|---|---|---|

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-14-b956190cd9d3> in <module>
     21
     22 #display quicklook - first image
---> 23 with zipfile(input_S1_files[0], 'r') as qck_look:
     24     qck_look=qck_look.open(name[0]+'.SAFE/preview/quick-look.png')
     25     img=mpimg.imread(qck_look)

IndexError: list index out of range
```

In [ ]:
```
from glob import iglob
```

In [ ]:
```
!pip install join
```

In [ ]:
```
import join
```

In [ ]:
```
import module
```

In [ ]:
```

```