**Exercise #3**

**This exercise will challenge you to code on your own. You will be given minimal direction and should think about how to figure out how to write the code to complete the task on your own. Feel free to use examples from the internet to help you build your code. If you get stuck, consult with Adam or Koreen!**

Challenge # 1

Take a look at the VIIRS 16 day Vegetation Products: ee.ImageCollection("NOAA/VIIRS/001/VNP09GA")

Pick an AOI and date range (e.g. the 2019 growing season at Victor). Cloud-mask VIIRS data for your chosen AOI and date range. Finally, pick a VIIRS band and extract the values from your AOI and plot them using pandas and matplotlib.

*(hint: remember to look at what bands this imagery contains, and what bits might be used to store cloud information. If you get stuck, there is a hint at the end of this document)*

Challenge # 2:

Use Sentinel-1 to display changes in water extent from the St. John NB flood in 2019. Which S1 mode should you use? How would you monitor the change in flood waters over time?

Challenge # 3:

Find some example Javascript code on the internet (e.g. in the developer’s guide, or on a forum somewhere) and convert it to python. First try to do this manually, but if you get stuck, try the automatic conversion script provided. Remember that with complex functions/code, the automatic converter does not always work perfectly!

**HINT for Challenge #1**

You should look at the documentation for this sensor to determine which bands to use. The link provided says that if pixel reliability is anything other than 0, 1 or 2 then it is a bad pixel.

Therefore, you can use set up your bitwise selection like this:

def maskVIIRSclouds(image):

qa = image.select('pixel\_reliability')

# if pixel reliability is anything other than 0, 1 or 2, its a bad pixel

cloud = qa.bitwiseAnd(1 << 3) \

.Or(qa.bitwiseAnd(1 << 4)) \

.Or(qa.bitwiseAnd(1 << 5)) \

.Or(qa.bitwiseAnd(1 << 6)) \

.Or(qa.bitwiseAnd(1 << 7)) \

.Or(qa.bitwiseAnd(1 << 8)) \

.Or(qa.bitwiseAnd(1 << 9)) \

.Or(qa.bitwiseAnd(1 << 10)) \

.Or(qa.bitwiseAnd(1 << 11))

# Remove edge pixels that don't occur in all bands

mask2 = image.mask().reduce(ee.Reducer.min())

return image.updateMask(cloud.Not()).updateMask(mask2)