Unlocking Insights from Above: A Journey into Remote Sensing with Python and Google Earth Engine

In today's data-driven era, the fusion of Python programming and Google Earth Engine presents an enticing entry point into the realm of remote sensing. This distinctive combination enables individuals to easily access satellite imagery and geospatial data, providing valuable insights across diverse domains such as environmental assessment, disaster response, precision agriculture, and urban planning. Throughout this project, I'll embark on a journey to unveil the complete capabilities of this tool, helping us to confront real-world challenges and achieve a deeper understanding of our changing planet.

In [ ]:
```
import ee
from IPython.display import Image

ee.Authenticate()
ee.Initialize()
```

To authorize access needed by Earth Engine, open the following URL in a web browser and follow the

https://code.earthengine.google.com/client-auth?
scopes=https%3A//www.googleapis.com/auth/earthengine%20https%3A//www.googleapis.com/aut
c&tc=pEZnFXx6DfYYev3lrvpRhJIesTCQTzfykAtEL5iYZ_Q&cc=RU2iyp7HZ4eMw0ZZG3J-suqANpQkD19

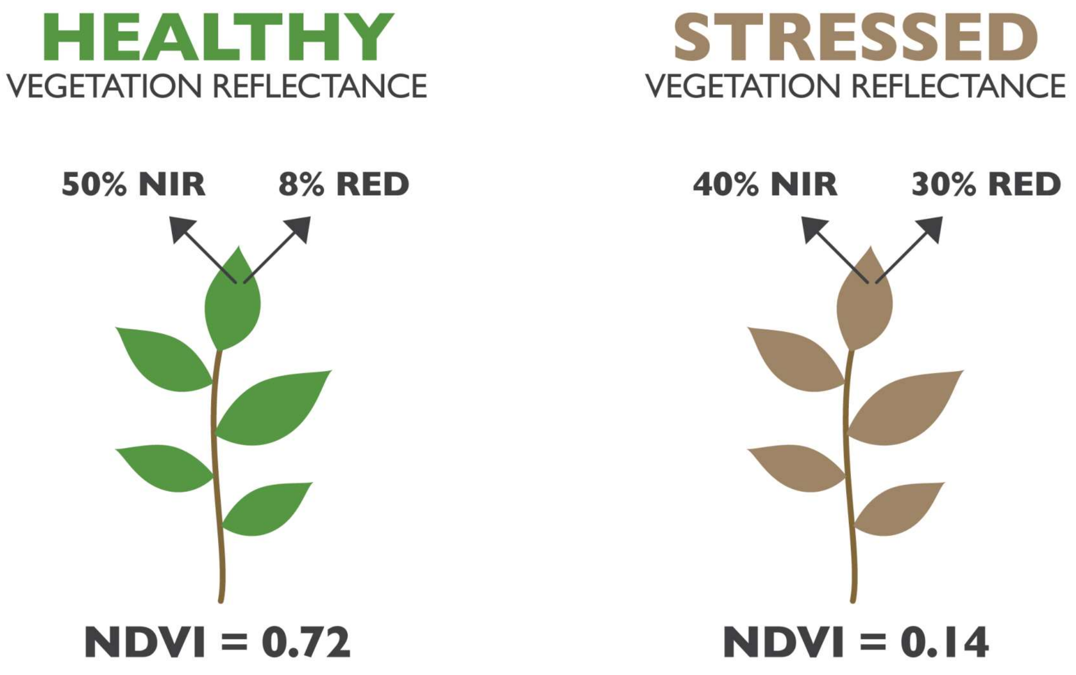
The authorization workflow will generate a code, which you should paste in the box below.

```
Successfully saved authorization token.
```

NDVI, which stands for Normalized Difference Vegetation Index, is a widely used vegetation index in remote sensing and environmental monitoring. It's calculated from satellite or aerial imagery and is used to assess and monitor the health and density of vegetation in a specific area. Here's how NDVI is calculated:

NDVI = (NIR - Red) / (NIR + Red)

NIR (Near-Infrared) is the reflectance in the near-infrared part of the electromagnetic spectrum. Red is the reflectance in the red part of the electromagnetic spectrum. In the case of Google Earth Landsat 8 data, the band SR_B5 corresponds to NIR and SR_B4 to Red. The NDVI values typically range from -1 to 1. Values close to -1 represent non-vegetated surfaces such as water bodies or barren land. On the other hand, values near 1 represent healthy and dense vegetation. As we take the average of the entire image, we have to note that water surfaces decrease the mean, and cloud coverage can also alter our results. Let's test it out in a real example.



$$\text{NDVI} = \frac{\text{NIR} - \text{RED}}{\text{NIR} + \text{RED}}$$

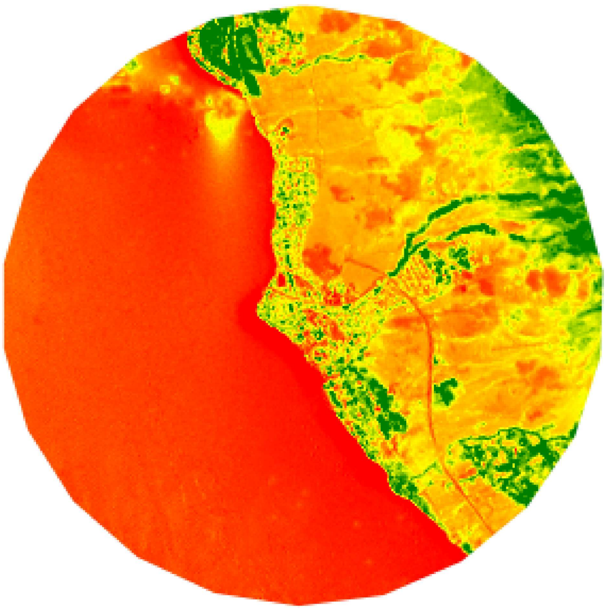
In early August 2023, a devastating series of wildfires erupted in the state of Hawaii, primarily affecting the island of Maui. These fast-spreading fires were driven by strong winds, leading to urgent evacuations, extensive destruction, and a tragic toll. The town of Lahaina, Hawaii, bore the brunt of the disaster, with at least 115 lives lost and over 1,000 individuals reported as missing.

To gain insights into the extent of this event, I wrote the following code to extract aerial imagery capturing the aftermath of these wildfires. Specifically, I selected two images, one taken before the incident and another captured after to assess the scale of the wildfire's spread and the severity of the damage inflicted upon the affected areas.

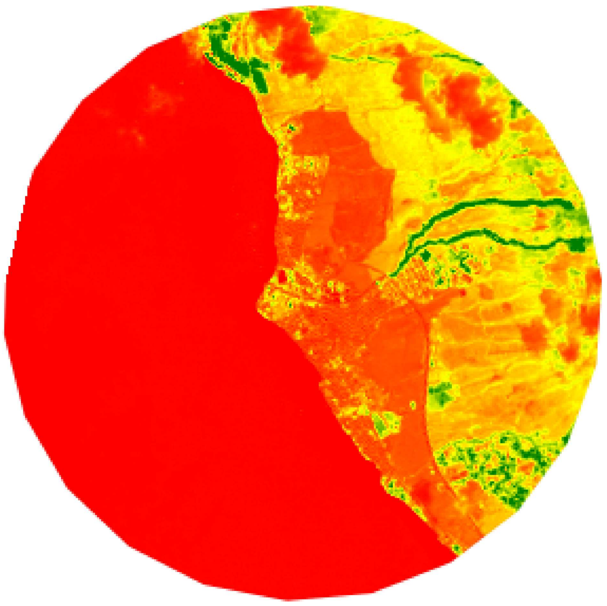
In [ ]:
```python
# Coordinates of Lahaina, Hawaii
lat = 20.883071
lon = -156.681378
poi = ee.Geometry.Point(lon,lat)
# Set time range
start_date = '2023-07-28'
end_date = '2023-08-31'
# Extract a collection of Landsat 8 images during that time and Location
landsat = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')\
    .filterBounds(poi)\
    .filterDate(start_date, end_date)
landsat = landsat.sort('system:time_start')
landsat_list = landsat.toList(landsat.size())
```

```python
# Indexes of the images I selected
landsat_sequence = [1,5]
# Define the region of interest (ROI) around the location
roi = poi.buffer(4250)

# Define the NDWI color palette
# Red is low vegetation, green is high
palette = ['red', 'yellow', 'green']
ndvi_parameters = {
    'min': 0.0,
    'max': 0.3,
    'palette': palette,
    'region': roi
}

# Print the total number of images in the collection
print('Total number of images:', landsat.size().getInfo())

# Extract data from the images and display them using the NDVI palette
for i in landsat_sequence:
    date = ee.Image(landsat_list.get(i)).get('DATE_ACQUIRED').getInfo()
    cloud = ee.Image(landsat_list.get(i)).get('CLOUD_COVER').getInfo()
    print('Date: ', date)
    print(f'Cloud Cover: {cloud}')
    ndvi = ee.Image(landsat_list.get(i)).normalizedDifference(['SR_B5', 'SR_B4'])
    ndvi_stats = ndvi.reduceRegion(reducer=ee.Reducer.mean(), geometry=roi, scale=30)
    mean_ndvi = ndvi_stats.get('nd').getInfo()
    print('Mean NDVI:', '{:.2f}'.format(mean_ndvi))
    display(Image(url=ee.Image(landsat_list.get(i)).normalizedDifference(['SR_B5', 'SF
```

```
Total number of images: 8
Date:  2023-07-29
Cloud Cover: 23.93
Mean NDVI: 0.09
```



```
Date:  2023-08-23
Cloud Cover: 16.28
Mean NDVI: 0.05
```



A noticeable distinction becomes apparent when we examine the images. The vegetation index has notably decreased by 0.04, a significant change, especially when we consider that nearly half of the image is covered by water, and the region in question is already relatively arid. The town lost nearly 45% of its vegetation. It becomes clear that the urban area has been severely affected by the fire, leaving no trace of vegetation. Additionally, we notice the presence of a road that served as an effective barrier, halting the wildfire's progression further into the island.

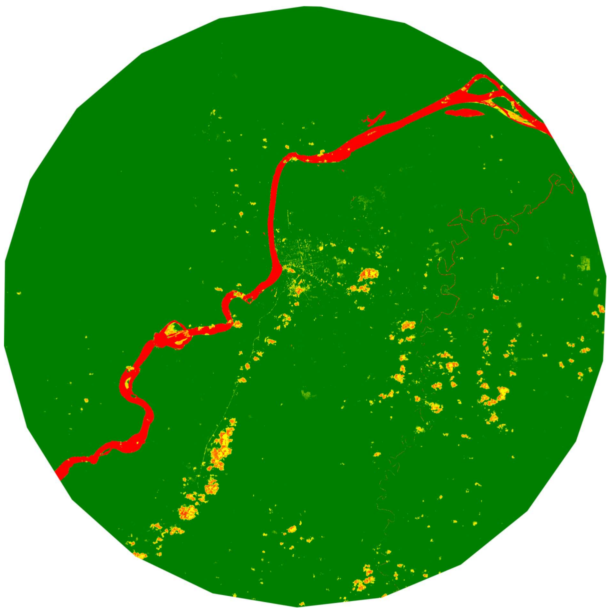
Patrick T. Fallon/AFP via Getty Images

Beyond its role in assessing wildfires, NDVI proves to be a valuable tool for examining a broader range of phenomena, including deforestation. The issue of deforestation in the Amazon rainforest has gained global attention due to its far-reaching implications. This vast and ecologically diverse region, spanning multiple South American nations, is grappling with the lasting impacts of human activities such as logging, agriculture, mining, and infrastructure development.

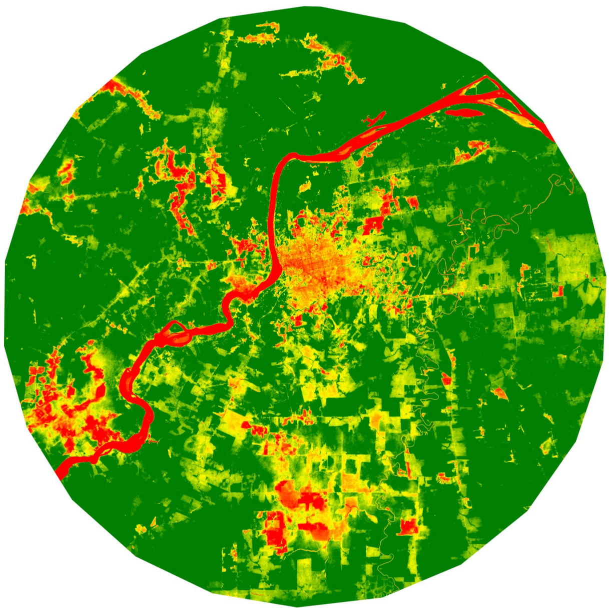
In this analysis, my focus shifts to Porto Velho, the capital city of Brazil's Rondônia state. Porto Velho boasts a diverse economy, including agriculture, livestock farming, mining, and timber production. Its strategic position as a transportation hub, connected by river and road networks to various parts of Brazil, underscores its importance in regional development. The goal is to understand the long-term effects of economic growth on the environment, particularly in terms of deforestation. To achieve this, I utilize Landsat 5 imagery spanning from 1986 to 2010. Again, I rely on straightforward indicators, such as NIR (band B4) and Red (band B3).

In [ ]:

```python
# Coordinates of Porto Velho, Brazil
lat = -8.814223
lon = -63.882731
poi = ee.Geometry.Point(lon, lat)
# Set time range
start_date = '1986-08-30'
end_date = '2010-08-30'
# Extract a collection of Landsat 5 images during that time and location
landsat = ee.ImageCollection('LANDSAT/LT05/C01/T1')\
    .filterBounds(poi)\
    .filterDate(start_date, end_date)
landsat = landsat.sort('system:time_start')
landsat_list = landsat.toList(landsat.size())
# Indexes of the images we want to select
landsat_sequence = [1,-1]
# Define the region of interest (ROI) around the location
roi = poi.buffer(40000)

# Define the NDWI color palette
# Red is low vegetation, green is high
ndvi_palette = ['red', 'yellow', 'green']
ndvi_parameters = {
    'min': -0.1,
    'max': 0.1,
    'palette': ndvi_palette,
    'region': roi,
}

# Print the total number of images in the collection
print('Total number of images:', landsat.size().getInfo())

# Extract data from the images and display them using the NDVI palette
for i in landsat_sequence:
    date = ee.Image(landsat_list.get(i)).get('DATE_ACQUIRED').getInfo()
    cloud = ee.Image(landsat_list.get(i)).get('CLOUD_COVER').getInfo()
    print('Date: ', date)
    print(f'Cloud Cover: {cloud}')
    ndvi = ee.Image(landsat_list.get(i)).normalizedDifference(['B4', 'B3'])
    ndvi_stats = ndvi.reduceRegion(reducer=ee.Reducer.mean(), geometry=roi, scale=30)
    mean_ndvi = ndvi_stats.get('nd').getInfo()
    print('Mean NDVI:', '{:.2f}'.format(mean_ndvi))
    display(Image(url=ee.Image(landsat_list.get(i)).normalizedDifference(['B4', 'B3'])
Total number of images: 239
Date:  1986-11-05
Cloud Cover: 6
Mean NDVI: 0.47
```



```
Date:  2010-08-19
Cloud Cover: 1
Mean NDVI: 0.11
```



From the maps, a clear transformation is evident in just a span of 24 years. The surrounding areas have lost nearly 75% of the original vegetation. The city center has experienced substantial expansion, accompanied by the construction of new roads. However, this development comes at a cost. Along these newly carved roads, we observe extensive deforestation and land clearing, primarily driven by the demands of agriculture and livestock farming. It's notable that the state of Rondonia in western Brazil has now emerged as one of the most heavily deforested regions within the vast expanse of the Amazon rainforest. This rapid alteration of the landscape underscores the complex interplay between economic growth and environmental impact, raising critical questions about the sustainability of such practices in the long run.

Just like with NDVI, we can use different bands provided by satellites to analyze other characteristics of the planet. The Earth's surface is mostly water, so learning about its behavior from the air can be of interest. NDWI, or Normalized Difference Water Index, is another valuable index used in remote sensing. It is specifically designed to highlight the presence and changes in water bodies. I found that NDWI can be calculated in multiple ways, implementing different bands, including NIR and SWIR, Green and SWIR, and Red and NIR. In this specific case, using Landsat 5 data, I found using Green and shortwave infrared (SWIR) bands to emphasize the best my goal:

NDWI = (Green - SWIR) / (Green + SWIR)

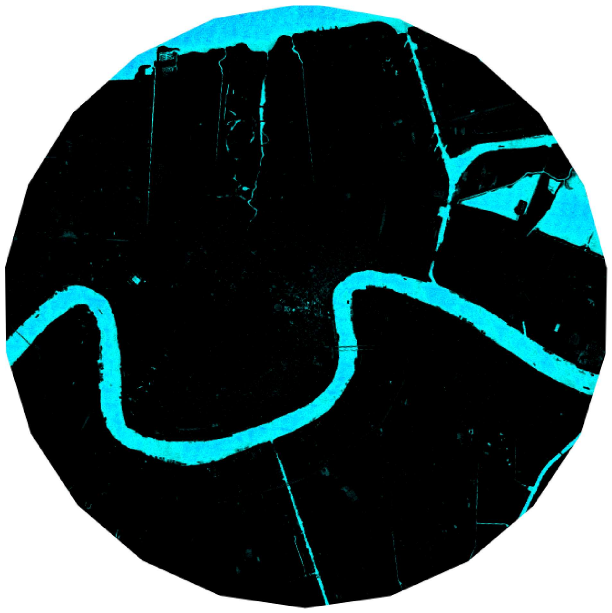
Green is the reflectance in the green part of the electromagnetic spectrum. SWIR represents the reflectance in the shortwave infrared part of the electromagnetic spectrum. In the case of Google Earth Landsat 5 data, you can use the bands B2 for Green and B5 for SWIR. Similar to NDVI, NDWI values typically range from -1 to 1. Values close to -1 indicate non-water features, while values near 1 represent water bodies.

NDWI proves to be a valuable tool for monitoring a wide range of surface events on our planet, with one prominent application being the detection of floods. Flooding is a recurring natural disaster that occurs worldwide, leading to significant damage and, tragically, loss of lives. To illustrate the utility of NDWI in this context, we can turn our attention to the devastating Hurricane Katrina that struck New Orleans. Hurricane Katrina, one of the most destructive hurricanes in U.S. history, made landfall in August 2005. It resulted in catastrophic flooding in New Orleans, Louisiana, and the surrounding areas. The flooding was particularly severe due to the city's unique geographical location, which is below sea level and relies on a complex system of levees and pumps to manage water levels.

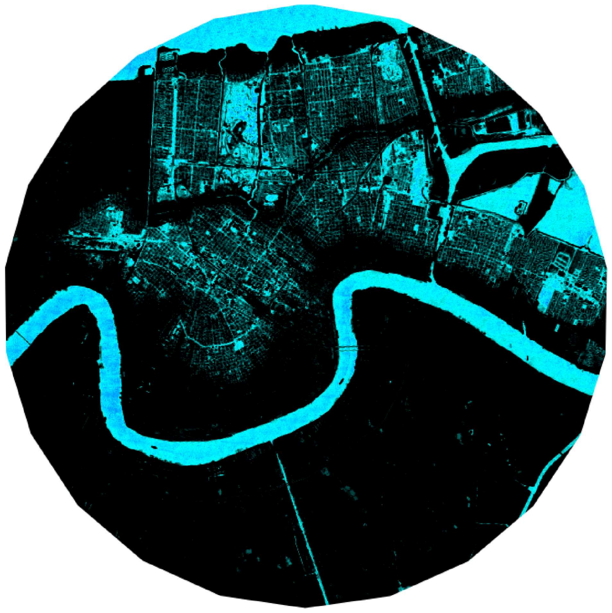
In [ ]:

```python
# Coordinates of New Orleans
lat = 29.951065
lon = -90.071533
poi = ee.Geometry.Point(lon, lat)
# Set time range
start_date = '2005-08-20'
end_date = '2005-09-30'
# Extract a collection of Landsat 5 images during that time and location
landsat = ee.ImageCollection('LANDSAT/LT05/C01/T1')\
    .filterBounds(poi)\
    .filterDate(start_date, end_date)
landsat = landsat.sort('system:time_start')
landsat_list = landsat.toList(landsat.size())
# Define the region of interest (ROI) around New Orleans
roi = poi.buffer(10000)

# Define the NDWI color palette
# Black represents non-water areas, blue represents water bodies
ndwi_palette = ['black', 'cyan', 'blue']
ndwi_parameters = {
    'min': -0.2,
    'max': 1.0,
    'palette': ndwi_palette,
    'region': roi,
}

# Print the total number of images in the collection
print('Total number of images:', landsat.size().getInfo())

# Extract data from the images and display them using the NDWI palette
for i in range(landsat.size().getInfo()):
    image = ee.Image(landsat_list.get(i))
    date = image.get('DATE_ACQUIRED').getInfo()
    cloud = image.get('CLOUD_COVER').getInfo()
    print('Date: ', date)
    print(f'Cloud Cover: {cloud}')
    ndwi = image.normalizedDifference(['B2', 'B5'])
    mean_ndwi = ndwi.reduceRegion(reducer=ee.Reducer.mean(), geometry=roi, scale=30).
    print('Mean NDWI:', '{:.2f}'.format(mean_ndwi))
    display(Image(url=image.normalizedDifference(['B2', 'B5']).getThumbUrl(ndwi_parame
```

```
Total number of images: 2
Date:  2005-08-22
Cloud Cover: 13
Mean NDWI: -0.23
```



```
Date:  2005-09-07
Cloud Cover: 3
Mean NDWI: -0.18
```

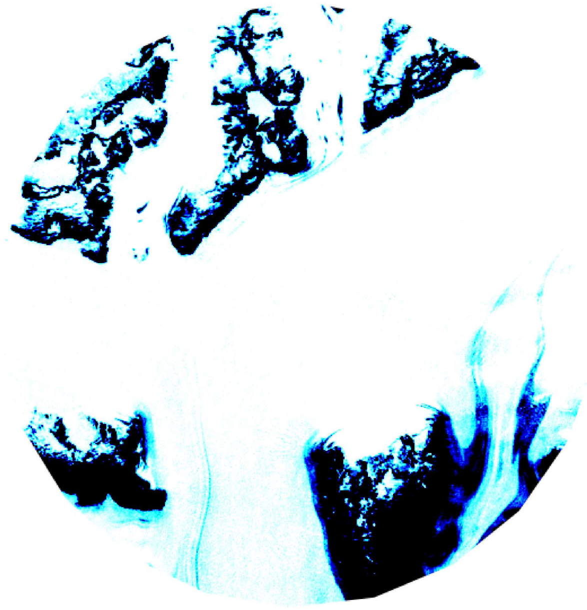


The images undeniably depict the profound extent of flooding that overwhelmed the City of New Orleans in the aftermath of Hurricane Katrina. From our analysis, we can deduce that about 20% of the city became underwater. Parks, streets, and residential neighborhoods are all submerged, conveying the gravity of the flooding event. Such visualization and analysis play a pivotal role in responding to disasters. This type of geospatial analysis yields crucial insights into the areas most severely impacted, enabling emergency responders and authorities to strategically deploy resources and aid efficiently. By pinpointing regions still immersed during the following weeks of the disaster, relief efforts can be tailored to provide essential assistance to affected residents, potentially saving lives and mitigating the disaster's consequences. It can also be useful to reconstruct better channel systems to avoid such destruction in the future.

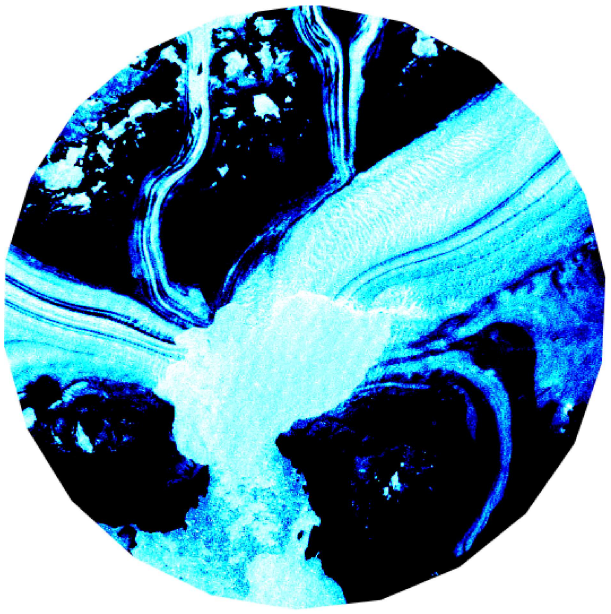
NDWI is a versatile index that has applications beyond flood monitoring. By adjusting the bands used in its calculation, we can derive other valuable indices like NDSI (Normalized Difference Snow Index) and NDII (Normalized Difference Ice Index). These indices are particularly useful for tracking changes in snow and ice cover, and some of the prominent locations of interest are glaciers. Monitoring glaciers using these indices allows us to gain insights into the impacts of global warming on snow melting and glacier recession. In the following code, I focus on the Columbia Glacier.

In [ ]:
```
# Define the coordinates for Columbia Glacier
lat = 61.150865
lon = -147.047084
poi = ee.Geometry.Point(lon, lat)
# Set time range
start_date = '1984-06-30'
end_date = '2010-08-30'
# Extract a collection of Landsat 5 images during that time and Location
landsat = ee.ImageCollection('LANDSAT/LT05/C01/T1')\
    .filterBounds(poi)\
    .filterDate(start_date, end_date)
landsat = landsat.sort('system:time_start')
landsat_list = landsat.toList(landsat.size())
# Indexes of the images we want to select
landsat_sequence = [0,-1]
# Define the region of interest (ROI) around the Location
roi = poi.buffer(8000)

# Define the NDSI color palette
# Black represents non-snow areas, white represents snow
ndsi_palette = ['black', 'blue', 'cyan', 'white']
ndsi_parameters = {
    'min': 0.4,
    'max': 0.9,
    'palette': ndsi_palette,
    'region': roi,
}

# Print the total number of images in the collection
print('Total number of images:', landsat.size().getInfo())

# Extract data from the images and display them using the NDSI palette
for i in landsat_sequence:
    image = ee.Image(landsat_list.get(i))
    date = image.get('DATE_ACQUIRED').getInfo()
    cloud = image.get('CLOUD_COVER').getInfo()
    print('Date: ', date)
    print(f'Cloud Cover: {cloud}')
    ndsi = image.normalizedDifference(['B2', 'B5'])
    mean_ndsi = ndsi.reduceRegion(reducer=ee.Reducer.mean(), geometry=roi, scale=30).g
    print('Mean NDSI:', '{:.2f}'.format(mean_ndsi))
    display(Image(url=image.normalizedDifference(['B2', 'B5']).getThumbUrl(ndsi_parame
```

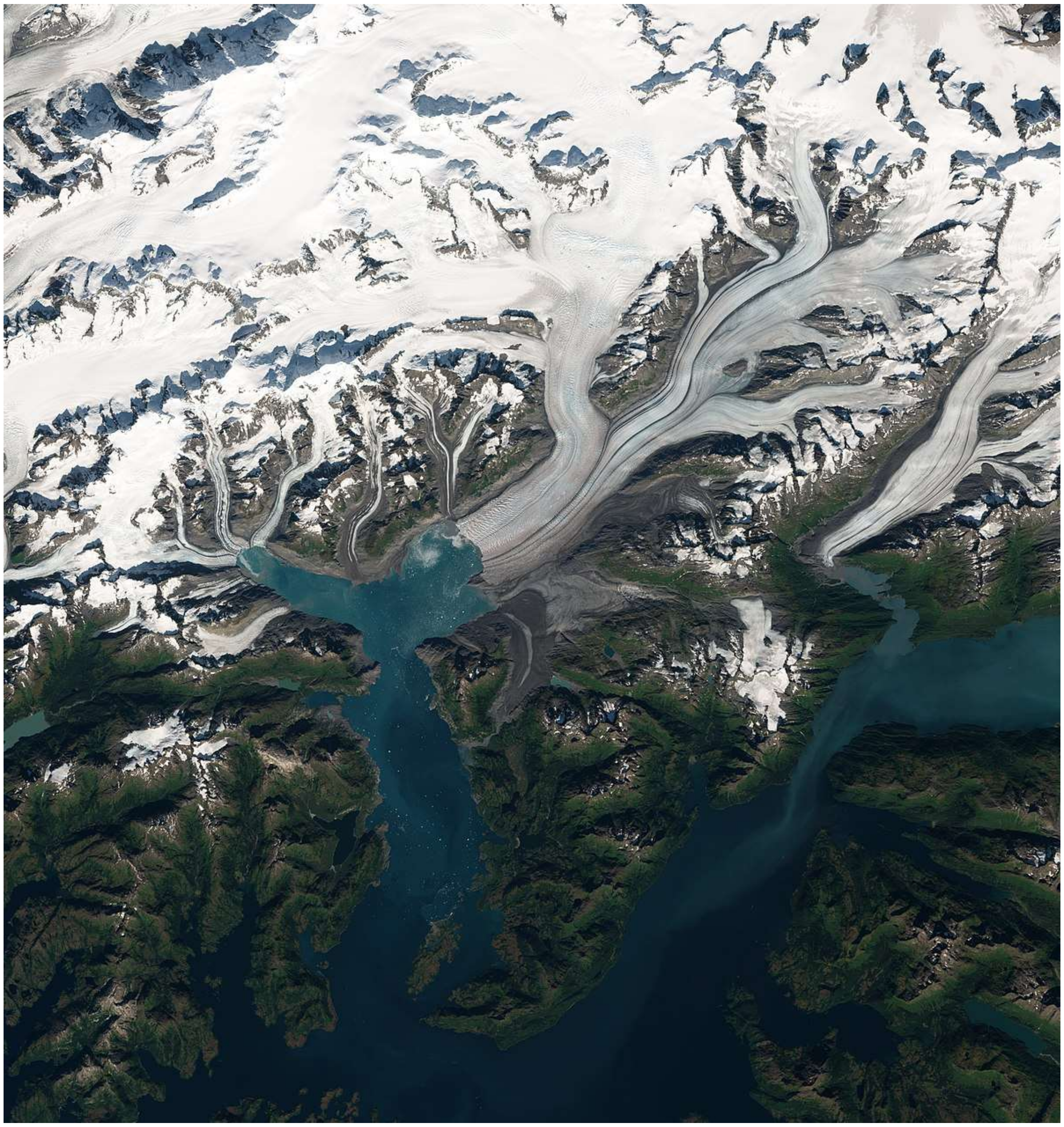
```
Total number of images: 261
Date:  1984-07-15
Cloud Cover: 31
Mean NDSI: 0.76
```



```
Date:  2010-08-24
Cloud Cover: 21
Mean NDSI: 0.42
```



The comparison between the two images taken roughly 26 years apart, both captured during the summer season, reveals a significant difference in snow cover. Nearly 45% of the glacier shown in the image has melted. This contrast highlights the unmistakable recession of the Columbia Glacier in Alaska. The darker tones in the images signify areas with no snow cover, further emphasizing the glacier's reduced volume. These changes in snow cover and glacier thickness serve as compelling evidence of the impact of global warming in this region. Less snow accumulates on the glacier during the winter months which contributes to higher levels of surface melt during the summer. These observations are in line with the well-documented phenomenon of glacier shrinkage worldwide, driven by rising global temperatures. The Columbia Glacier's recession is not only visually apparent but also indicative of a broader trend of glacier loss, which has far-reaching consequences for sea level rise, ecosystem dynamics, and regional climate patterns. Here's a more current high-quality image of the glacier extracted from Google Earth.



In conclusion, this Python and Google Earth Engine project has successfully demonstrated the power of combining advanced geospatial analysis with a versatile programming language. Through this project, I have learned the capabilities of Google Earth Engine to access and process large-scale Earth observation data, allowing me to perform complex analyses and generate valuable insights for various applications, particularly in the fields of environmental assessment and natural disaster monitoring.